ENCICLOPEDIA PRACTICA DE LA

INFORMATICA APLICADA

1

Cómo construir juegos de aventura

Manuel Alfonseca

SINCLAIR
AMSTRAD
COMMODORE
IBM



EDICIONES SIGLO CULTURAL

ENCICLOPEDIA PRACTICA DE LA

INFORMATICA APLICADA

1

Cómo construir juegos de aventura

EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática
JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño y maquetación:
BRAVO-LOFISH.

Dibujos:
JOSE OCHOA Y ANTONIO PERERA.

Tomo 1. **Cómo construir juegos de aventura**
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:
Sor Angela de la Cruz, 24-7.º G. Teléf. 279 40 36. 28020 Madrid.

Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28020 Madrid.

Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América. 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-019-7
ISBN de la obra: 84-7688-018-9.

Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.

Imprime:
TRAIMSA. Nicolás Morales, 38-40. 28019 Madrid.

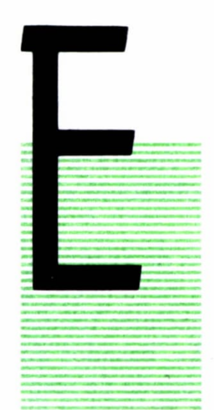
Depósito legal: M-32.946-1986
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Sor Angela de la Cruz, 24-7.º G.
Teléf. 279 40 36.

Octubre, 1986.
P.V.P. Canarias: 365 pts.

INDICE

JUEGOS DE AVENTURA 1



L auge actual de los juegos de computadora es una consecuencia lógica de la aparición del ordenador personal, y una de las causas principales de la impresionante proliferación de estas máquinas durante la década en que vivimos. En efecto, uno de los argumentos más importantes con los que el ordenador llega a introducirse en el hogar es la posibilidad de utilizarlo para distraer los ratos de ocio jugando. Y esto no se aplica sólo a los niños y a los jóvenes, pues también las personas mayores sucumben a menudo a la tentación de pasar algunas horas probando suerte en juegos de azar o de habilidad, o introduciéndose en un mundo de aventuras ficticias, que les permite pasar un rato de emociones violentas sin correr riesgos reales para su integridad física.

El número y variedad de juegos que ahora existen es enorme. Pero no es difícil clasificarlos en unos cuantos grupos: por un lado, de acuerdo con la forma de realizarse la interacción entre la máquina y el jugador, podemos dividirlos en juegos gráficos (es decir, basados en la presentación de imágenes) o discursivos (que se desarrollan a través de un diálogo entre el jugador y el ordenador). Por otra parte, de acuerdo con el tema del juego, podemos distinguir los siguientes grupos:

1. Juegos inteligentes. Muchos de ellos son clásicos: el ajedrez, las damas, el chaquete o «Backgammon», el «Go», el «Nim», el «Awari», las tres en raya, el «Mastermind». Otros son más recientes, como el «Hexapawn», inventado por Martin Gardner en 1962.

2. Juegos de azar. Pueden ser juegos de cartas simulados, como el póquer o el «Blackjack», versión inglesa de las siete y media, o bien juegos de casino, como la ruleta, los dados o las máquinas tragaperras.

3. Juegos de habilidad, que exigen que el jugador desarrolle una técnica especial, normalmente en el uso de los dedos para presionar ciertas

teclas. Los más conocidos son las diversas versiones de «comecocos». También son curiosas las simulaciones de deportes, como el golf, el decatlón, el tenis normal o de mesa, etc. Muchos de estos juegos utilizan al máximo las posibilidades gráficas de los ordenadores personales y son verdaderamente espectaculares.

4. Juegos educativos, cuyo objetivo es conseguir que el jugador aprenda algo. Pueden clasificarse, a su vez, en juegos de preguntas y respuestas (como los que permiten practicar operaciones aritméticas, o los que examinan los conocimientos del jugador sobre temas de literatura, geografía, historia, etc.); juegos de adivinar (números, letras, palabras, como el famoso «Ahorcado»), y juegos de simulación, que se utilizan también en situaciones reales, como los simuladores de vuelo de las academias de pilotos, los simuladores de conducción de submarinos y otros. Mención especial merecen los simuladores de procesos sociales (como «Hammurabi») que ponen al jugador al frente de un país y le obligan a darse cuenta, a través de las consecuencias de sus actos, de la increíble complicación e interrelación de nuestras sociedades.

5. Finalmente, y éste es el objeto de este libro, los juegos de aventura, de los que existen varias familias (aunque algunos pueden pertenecer a dos clases a la vez): simuladores de combate; aventuras espaciales (como el «Desembarco lunar» o los juegos de la familia «Star trek», basados en una conocida serie de televisión); por último, juegos de búsqueda de tesoros, de ambiente medieval, en los que el jugador debe «explorar» un territorio desconocido, vencer las dificultades que encuentre a su paso, apoderarse de los objetos que descubra y vencer a los enemigos más o menos fabulosos (dragones, brujas, vampiros, fieras, centauros, etc.) que le atacan. En los juegos más complejos de este tipo, estos enemigos pueden ser móviles. Además, cada uno de los objetos tiene una utilidad determinada: existen alimentos, armas, encantamientos, pócimas que se pueden beber y producen diversos efectos, y así sucesivamente.

El deseo de aventuras parece ser natural en el hombre. Además, se trata de uno de nuestros impulsos más importantes, pues es una de las causas principales de que nos sintamos empujados a investigar el ambiente que nos rodea, y a no cejar hasta dominarlo en lo posible. Es decir, que la evolución cultural y los avances científicos de los que estamos tan orgullosos se deben, en parte no pequeña, a nuestras tendencias aventureras.

El hombre primitivo compartía, evidentemente, este tipo de impulsos (pues, en caso contrario, jamás habría surgido la civilización), pero a él no le faltaban ocasiones de ponerlo en práctica: su vida era una lucha continua contra un ambiente hostil. Sin embargo, con la Revolución Neolítica, que tuvo lugar hace unos ocho o diez mil años, las circunstancias cambiaron. A partir de la invención de la Agricultura, la sociedad humana se especializó. En efecto, con las nuevas técnicas agrícolas, un hombre solo era

capaz de producir alimentos para gran número de personas, lo que hacía innecesario que todas ellas se dedicaran al mismo oficio. Algunos, por tanto, inventaron la cerámica y cambiaban los productos de su arte por los alimentos que le sobraban al agricultor. Otros, los más fuertes, le ofrecieron sus servicios a cambio de comida para defenderle de los ataques de vecinos envidiosos. Unos pocos construían instrumentos de labranza o armas de guerra, y aprendieron a obtener y trabajar los metales.

La abundancia de alimentos provocada por la nueva situación dio lugar a un aumento rápido de la población. Las antiguas tribus, formadas por algunas decenas de individuos que controlaban un territorio relativamente vasto, se vieron suplantadas por la acumulación de gran número de personas en un espacio reducido. El estilo de vida y de vivienda cambió también, y surgió la ciudad.

Con la aparición de la civilización urbana, el hombre común encontró menos posibilidades para satisfacer su impulso de aventuras. Es verdad que las catástrofes naturales (inundaciones, erupciones volcánicas, terremotos, desprendimientos de tierra) seguían existiendo. Además, las catástrofes humanas (invasiones de pueblos vecinos, guerras de conquista) eran ahora incomparablemente más violentas que en la antigüedad, pues afectaban a un número mucho mayor de personas. Sin embargo, se trataba de sucesos cuya frecuencia no era grande. No era probable que una de estas catástrofes afectara a un ser humano más de una vez en la vida, excepto en el caso de que perteneciera a la casta militar, que tenía la guerra como profesión y medio de vida. Había también algunas personas (los comerciantes y buscadores de fortuna) que a menudo realizaban largos viajes, conocían países lejanos y encontraban aventuras sin cuento a su paso, pero fueron siempre muy pocos. En consecuencia, la inmensa mayor parte de la población hubo de buscarse otros medios de satisfacer su deseo de emociones. Y, puesto que ellos no podían vivirlas, se conformaron con el sucedáneo de gozarlas vicariamente, escuchando las aventuras de los demás.

De siempre se sabe que la llegada de un viajero que viene de lejanas tierras despierta la atención de todos y congrega en derredor suyo a viejos y jóvenes, que desean oír sus relatos. O, al menos, así sucedía antes de que el impresionante desarrollo moderno de las comunicaciones y la facilidad de viajar de que hoy disfrutamos eliminaran para siempre el misterio de los países extranjeros y exóticos.

Había también otros medios: las historias y aventuras no tenían necesariamente que ser verídicas. Y así surgieron por doquier hombres que se especializaban en contar y cantar las hazañas de los grandes héroes del pasado reciente, lejano o ficticio, hazañas que se iban adornando al pasar de boca en boca y que, si tenían una base real, acababan por parecerse muy poco a lo que en verdad había sucedido. Los cantores, los juglares, desempeñaron un papel muy importante en una sociedad humana cuyos miembros habían perdido la posibilidad de vivir sus propias emociones.

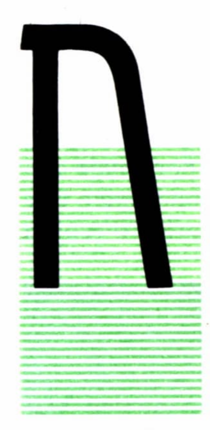
Pero los relatos de los cantores tenían una duración muy breve. Tan sólo podía disfrutarse de ellos durante la actuación de su autor o, como mucho, entre las nebulosidades de la memoria. Con la invención de la escritura, que tuvo lugar hace unos cinco mil años, fue posible conservarlos y releerlos a placer. Y algunos de ellos llegaron a convertirse en obras maestras de la literatura universal: el poema de Gilgamesh, la historia de Sinuhé el egipcio y de Simbad el marino, la *Ilíada, La Odisea...*

Es verdad que la posibilidad de leer quedó fuera del alcance de la mayoría de las personas durante miles de años, pero al menos la forma de las grandes obras quedó fija y pudo conservarse para deleite de generaciones futuras. Y a partir de la invención de la imprenta en el siglo XI en China, y en el XV en la civilización occidental, junto con la extensión creciente de la educación básica, el número de lectores en potencia se multiplicó. La lectura directa de las aventuras y hazañas de los demás se convirtió, por tanto, en la forma principal de satisfacer nuestro deseo de emociones. No es extraño, por consiguiente, que el siglo XVI sea precisamente (en Occidente) la época en que comienza a proliferar un nuevo género literario, la novela, que se dirige especialmente a cumplir ese papel.

En los últimos siglos hemos sido testigos de una nueva revolución social, comparable a la provocada por la invención de la Agricultura. La revolución industrial ha aumentado otra vez extraordinariamente los recursos humanos, provocando, en consecuencia, un nuevo aumento desmesurado de la población. El estilo de vida ha cambiado y la ciudad antigua y medieval ha dejado paso a las megalópolis actuales, donde se hacinan muchos millones de habitantes. También en esta ocasión los avances de la técnica nos han proporcionado formas nuevas de satisfacer el deseo de emociones: los diversos medios de grabación de la imagen (cinematógrafo, televisión, vídeo) nos permiten ahora no sólo oír las aventuras de los demás, sino también verlas. Es un avance, pero el gozo obtenido sigue siendo un sucedáneo.

Sin embargo, la aparición de los ordenadores, con sus posibilidades de simular situaciones ficticias, pero semejantes a las reales, e interactivas (es decir, con participación humana), ha puesto a nuestro alcance, por primera vez, la posibilidad de vivir nuestras propias aventuras emocionantes, librándonos, por otra parte, del riesgo que tendríamos que correr si éstas fueran verdaderas. Esta es, seguramente, la causa del gran éxito actual de los juegos de aventura.

JUEGOS DE GUERRA 2



UESTRA época está llena de contradicciones y paradojas. Al mismo tiempo que los medios de comunicación expresan opiniones y publican artículos de psicólogos y educadores en contra de los llamados «juguetes bélicos» y sus supuestos efectos nocivos sobre la mentalidad infantil, algunos de estos mismos medios (especialmente la televisión) anuncian o exhiben películas y programas cuyo grado de violencia es extremo, poniéndolas al alcance de estos mismos niños a los que, por otro lado, afirman proteger.

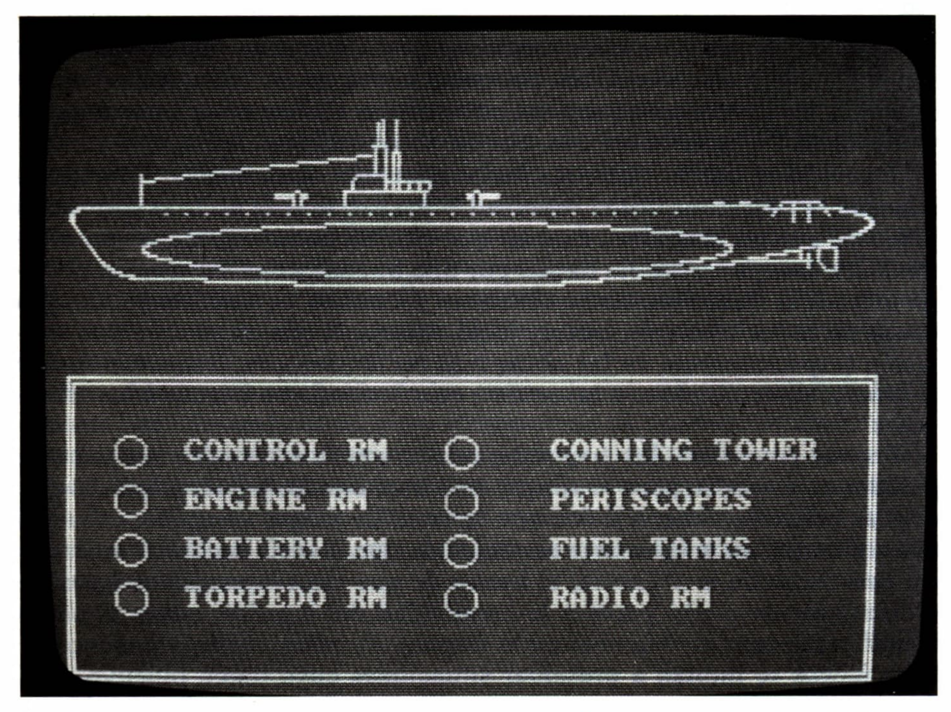
Es evidente que la violencia real o simulada que entra por los ojos, y a la que se reacciona como espectador pasivo, es mucho más dañosa en potencia que un juego inocente en que el niño pone en práctica su creatividad, e imagina estar luchando por una buena causa. Porque en el juego infantil la distinción entre «buenos» y «malos» es siempre nítida, y la causa de la lucha es una razón moral. Mientras que en el cine y la televisión (especialmente en las obras más recientes) esta distinción tiende a difuminarse, o incluso a invertirse, convirtiendo en héroes a los criminales.

En este panorama, los juegos de aventura para ordenador ocupan una posición intermedia, pero se alinean más bien con los juguetes bélicos que con las películas violentas. Aunque se trata de juegos «dirigidos», puesto que contienen un programa preestablecido que maneja las líneas maestras del desarrollo de la aventura, el hecho de que sean interactivos concede ordinariamente cierto papel a la creatividad del jugador, quien tiende a identificarse con los «buenos» en la disputa. Al mismo tiempo, muchos de estos juegos se limitan a presentar situaciones cuyo único objetivo es la destrucción del mayor número de enemigos posible, antes de que el jugador sea, a su vez, «vencido». En estos casos, el juego se convierte en un mero concurso de habilidad con los dedos, al que tal vez sería más educativo despojar de sus connotaciones bélicas. Pero, lamentablemente, pa-

rece que este componente violento ayuda en verdad a hacerlos más interesantes y atractivos para el público.

Existe una película de bastante éxito, que lleva el mismo título de este capítulo, y cuyo objetivo primordial (aparte del evidente de ganar dinero para la compañía productora) es denunciar públicamente los peligros de la «adicción» de muchos jóvenes a los juegos bélicos de ordenador, que les puede llevar a faltar incluso a la ley. Es, en efecto, cierto que ha habido casos reales de muchachos que lograron introducirse en sistemas de ordenador que, en pura lógica, deberían estarles vedados, y que han logrado extraer de ellos datos confidenciales. Algunos de estos casos han sido descubiertos y han llegado a los tribunales, impulsando además al ejército y a la gran banca norteamericana a establecer sistemas de palabras clave más seguros e indescifrables.

Los guionistas de la película en cuestión no pudieron resistir la tentación de introducir en ella el típico «computador maligno», personaje clásico de las peores novelas de ficción científica, el esclavo tecnológico que se niega a obedecer a sus amos humanos y trata de dominarlos o (como



Los juegos de guerra se convierten a veces en verdaderas pruebas de habilidad. El juego «Gato», por ejemplo, simula la sala de control de un submarino. El jugador debe dirigir los mandos y mover la nave correctamente, eludiendo los ataques de los barcos enemigos y hundiendo el mayor número posible de éstos.

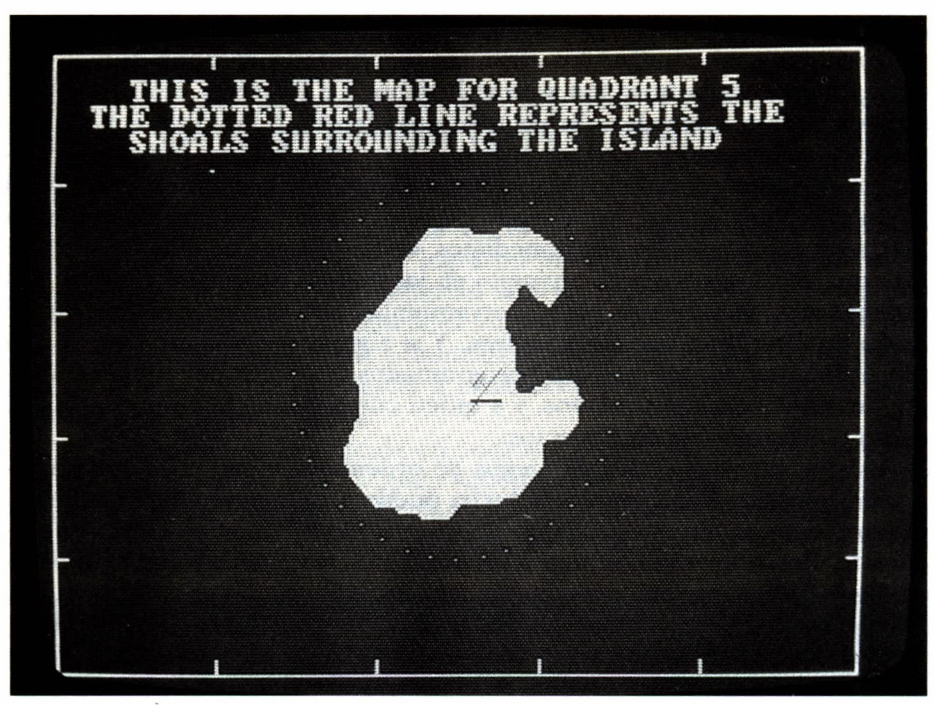
en este caso) de ser más listo que ellos. Ignoro si algún día se llegará a construir máquinas que tengan los atributos de libertad y responsabilidad que normalmente atribuimos tan sólo al hombre y a los seres espirituales. Si tal cosa llegara a suceder, dichas máquinas tendrían tanto derecho como nosotros a ser consideradas «humanas», provistas de alma inmortal, y posiblemente se produciría un nuevo movimiento por la emancipación de la esclavitud. Pero de algo estoy seguro: que ni los actuales pobladores de la Tierra, ni sus descendientes inmediatos, serán testigos de ello. Se trata de algo que, si es posible, está envuelto en las nieblas del futuro remoto. Los ordenadores de hoy son puras herramientas tan desprovistas de mente como un martillo o una llave inglesa. Y si alguna vez realizan una acción reprobable, es porque han sido programados para ello por un ser humano, de quien son instrumentos, y a quien corresponde la responsabilidad de las consecuencias de sus actos.

Existen diversas clases de juegos de ordenador basados en simulaciones de combate. Por ejemplo, la lucha puede presentarse en forma de enfrentamiento de naves espaciales, de los que hablaremos en el capítulo siguiente. Restringiéndonos aquí a los combates realizados sobre la superficie de la Tierra, podemos distinguir, por un lado, los juegos de habilidad, que se basan en la eliminación del mayor número posible de enemigos, que van apareciendo de modo aleatorio y que, o bien tratan de destruir al jugador (que tendrá que eludirlos), o bien son totalmente pasivos, concediéndosele en este caso un tiempo determinado para conseguir el objetivo del juego. En segundo lugar, los juegos bélicos educativos, que tratan de demostrar alguna propiedad física (como la dependencia entre la trayectoria de una bala de cañón y el ángulo de tiro). Por último, las simulaciones de combates reales, con personajes históricos concretos, uno de los cuales es representado por el jugador, que trata de conseguir la victoria (o evitar la derrota) que logró o sufrió su famoso antecesor.



UN JUEGO BELICO EDUCATIVO

Uno de los juegos más sencillos y conocidos, del que existen numerosas versiones, tiene por objeto alcanzar y destruir un objetivo enemigo mediante uno o varios disparos de cañón. Inicialmente conocemos el alcance máximo del arma y sabemos que ésta puede dispararse con distintos ángulos de tiro. Para simplificar el problema, supondremos que el cañón apunta directamente hacia el objetivo, por lo que el proyectil se moverá siempre sobre la línea que une ambos puntos, y sólo existirán tres posibilidades: que caiga lo suficientemente cerca del objetivo como para que podamos considerarlo destruido, que pase de largo, o que se quede corto. Este juego, que permite cinco intentos en cada caso concreto, enseña al



Muchos juegos llevan gráficos espectaculares, a veces en color, otras en blanco y negro. Esta fotografía corresponde al juego «Gato» y representa un mapa donde figura la posición del submarino.

jugador a relacionar el ángulo de disparo del cañón con la distancia alcanzada por el proyectil.

El programa 2.1 presenta la realización de este juego en el lenguaje BASIC.

Programa 2.1: Juego del disparo de cañón en lenguaje BASIC.

Este programa es válido para SPECTRUM, AMSTRAD, COMMODORE e IBM PC o compatibles (ver notas).

```
1 REM Juego del disparo de cañón. Por M. Alfonseca.
10 RANDOMIZE
30 LET A=10000+INT(RND*40000)
40 FOR N=1 TO 3
50 PRINT "El alcance de tu cañón es igual a ";A;" metros."
60 LET D=INT(RND*A)
70 PRINT "Hay un objetivo enemigo a una distancia de ";D;" metros."
80 FOR I=1 TO 5
90 PRINT "Dame el ángulo de elevación de tu cañón, en grados."
100 INPUT E
```

```
110 IF E<.5 THEN PRINT "El ángulo de elevación no puede ser menor que
0.5 grados": GOTO 90
120 IF E>89.5 THEN PRINT "El ángulo de elevación no puede pasar de
89.5 grados": GOTO 90
130 LET R=INT(A*SIN(2*E*3.1415927/180))
140 IF 100 >= ABS(R-D) THEN GOTO 210
150 IF R>D THEN PRINT "Tu tiro se pasó en ";R-D;" metros."
160 IF R<D THEN PRINT "Tu tiro se quedó corto en ";D-R;" metros."
170 PRINT "Prueba otra vez."
180 NEXT I
190 PRINT "Lo siento. Tu unidad ha sido destruida por el enemigo."
200 GOTO 240
210 PRINT "El objetivo enemigo saltó por los aires."
220 PRINT "Lo has logrado en ";I;" intentos."
230 NEXT N
240 PRINT "Gracias por jugar conmigo."
```

NOTAS: El programa anterior es válido directamente para SPECTRUM e IBM PC. En algunos intérpretes BASIC, la palabra reservada LET no es necesaria, pero también es reconocida si se utiliza.

Para AMSTRAD cambiar las siguientes líneas:

```
30 LET A=10000+INT(RND(1)*40000)
60 LET D=INT(RND(1)*A)
```

Para COMMODORE cambiar las siguientes líneas:

```
10 LET M=RND(-TI)
30 LET A=10000+INT(RND(1)*40000)
60 LET D=INT(RND(1)*A)
```

Programa 2.1. *Juego del «Disparo de cañón», en lenguaje BASIC.*

Veamos en qué consiste este programa.

1. La línea 1 es un comentario que contiene el nombre y autor del programa.

2. La línea 10 inicializa el generador de números aleatorios del intérprete de BASIC. Esta instrucción puede eliminarse sin que el programa sufra, aunque en tal caso los números aleatorios generados durante el resto del programa serán siempre los mismos, cada vez que se ejecute el juego. Si se utiliza, en algunos ordenadores el programa pedirá al jugador que le especifique un número entero cualquiera, que utilizará para inicializar el generador de números aleatorios.

3. La línea 20 le dice al intérprete de BASIC que las variables cuyos nombres comienzan por las letras A, D, E y R deben considerarse como variables decimales de precisión sencilla. En muchos intérpretes de BASIC esta instrucción puede eliminarse, pues los valores de las variables se considerarían de este tipo en cualquier caso. (Es decir, ésta es la opción por defecto.)

4. La línea 30 genera el alcance del cañón en metros, que se guardará en la variable A, y que resulta ser un número entero aleatorio comprendido entre 10.000 y 50.000.

5. La línea 40 comienza un bucle que se realiza tres veces, y que termina en la línea 230. Cada una de estas veces corresponde a uno de los tres objetivos que se le presentan a nuestro cañón, y que tendrá que destruir.

6. La línea 50 se limita a escribir en la pantalla el alcance de nuestro cañón.

7. La línea 60 calcula la distancia del objetivo como una proporción aleatoria del alcance del cañón (es decir, un número entero de metros comprendido entre cero y dicho alcance).

8. La línea 70 escribe en la pantalla la distancia del objetivo.

9. La línea 80 comienza un segundo bucle que se realiza cinco veces, y que termina en la línea 180. Cada una de estas veces corresponde a un intento para destruir el objetivo.

10. La línea 90 pide en la pantalla el ángulo de elevación que el jugador desea dar al cañón para realizar el próximo disparo.

11. La línea 100 lee del teclado el valor proporcionado por el jugador y lo guarda en la variable E.

12. La línea 110 comprueba que dicho ángulo no es menor que medio grado. Si lo es, lo rechaza con el mensaje correspondiente y vuelve a pedir el valor del ángulo (GOTO 90).

13. La línea 120 comprueba que dicho ángulo no es mayor que el valor máximo admitido (89,5 grados). Si lo es, lo rechaza con el mensaje correspondiente y vuelve a pedir el valor del ángulo (GOTO 90).

14. La línea 130 calcula la distancia en metros a la que cae el proyectil, y la guarda en la variable R. Esta distancia se calcula multiplicando el alcance del cañón por el seno del doble del ángulo de elevación, expresado en radianes. Para convertir el ángulo en radianes se multiplica su valor en grados por el valor de pi (3,1415927) y se divide por 180.

15. La línea 140 comprueba si la distancia alcanzada por el proyectil se diferencia de la posición del objetivo en menos de 100 metros en am-

bos sentidos. En tal caso, el objetivo se considerará destruido, y el programa pasa a ejecutar la instrucción 210.

16. Si la comprobación anterior no resultó válida, el programa pasa a ejecutar la línea 150, que comprueba si la distancia alcanzada por la bala del cañón ha rebasado el objetivo y, en tal caso, escribe un mensaje que indica en cuántos metros se ha pasado.

17. La línea 160 comprueba si la distancia alcanzada por la bala del cañón no ha llegado a alcanzar el objetivo y, en tal caso, escribe un mensaje que indica en cuántos metros se ha quedado corta.

18. La línea 170 escribe simplemente el mensaje «Prueba otra vez».

19. La línea 180 señala el final del bucle interno (el de los cinco intentos de destrucción del objetivo).

20. Si no hemos conseguido destruirlo en cinco intentos o menos, el programa pasará a ejecutar la línea 190, que indica que hemos fracasado, pues el enemigo nos ha destruido antes de que consiguiéramos acabar con él.

21. La línea 200 salta hasta la línea 240 (GOTO 240) para dar por terminado el programa. Hemos perdido.

22. La línea 210 recibe control cuando nuestro disparo consiguió acercarse al objetivo lo suficiente para destruirlo. Esta línea se limita a escribir en la pantalla el mensaje correspondiente.

23. La línea 220 nos recuerda cuántos intentos hemos necesitado para destruir el objetivo.

24. La línea 230 señala el final del bucle externo (el de los tres objetivos que tenemos que destruir con el cañón).

25. Finalmente, la línea 240 (a la que se llega si hemos conseguido aniquilar los tres objetivos, o si nuestra unidad fue destruida por el enemigo) escribe en la pantalla una línea de despedida. Es el final del programa.

A continuación, el programa 2.2 presenta la realización de este juego en el lenguaje APL.

Programa 2.2: Juego del disparo de cañón en lenguaje APL.

Válido para cualquier máquina que disponga de un intérprete de APL.

```
[0] DISPARO
[1] A←10000+?40000
[2] N←1
[3] L:'El alcance de tu cañón es igual a ',(⍕A),' metros.'
[4] D←(?100)×A÷100
```

```
[5] I←1
[6] 'Hay un objetivo enemigo a una distancia de ',(⍕D),' metros.'
[7] L1:'Dame el ángulo de elevación de tu cañón, en grados.'
[8] E←⎕
[9] →(E<0.5)/E1
[10] →(E>89.5)/E2
[11] R←A×1○2×○E÷180
[12] →(100≥|R-D)/BIEN
[13] →(R>D)/MAS
[14] MENOS:'Tu tiro se quedó corto en ',(⍕D-R),' metros.'
[15] →OTRA
[16] MAS:'Tu tiro se pasó en ',(⍕R-D),' metros.'
[17] OTRA:→(5<I←I+1)/FIN
[18] 'Prueba otra vez.'
[19] →L1
[20] BIEN:'El objetivo enemigo saltó por los aires.'
[21] 'Lo has logrado en ',(⍕I),' intentos.'
[22] →(3≥N←N+1)/L
[23] 'Gracias por jugar conmigo.'
[24] →0
[25] E1:'El ángulo de elevación no puede ser menor que 0.5 grados.'
[26] →L1
[27] E2:'El ángulo de elevación no puede ser mayor que 89.5 grados.'
[28] →L1
[29] FIN:'Lo siento. Tu unidad ha sido destruida por el enemigo.'
```

Programa 2.2. *Juego del «Disparo de cañón», en lenguaje APL.*

El programa sigue exactamente los mismos pasos que la versión en BASIC, con las siguientes diferencias:

1. En APL no es necesario inicializar el generador de números aleatorios (aunque puede hacerse). Sucesivas pasadas de este programa darán lugar, automáticamente, a la generación de números aleatorios distintos.

2. No es necesario declarar ninguna variable. Podemos olvidarnos de si éstas van a contener números enteros, en precisión simple o doble. El intérprete se ocupa de decidirlo.

3. Es posible dar nombre a las instrucciones mediante etiquetas. Las líneas 3, 7, 14, 16, 17, 20, 25, 27 y 29 tienen etiqueta. Una etiqueta es un nombre colocado a la izquierda de la instrucción y separado de ella por dos puntos (:).

4. El nombre de una variable separado de un valor por una flecha hacia la izquierda significa que dicho valor queda asignado a la variable.

5. En APL no existen instrucciones condicionales explícitas de la forma «IF condición THEN instrucción». Las instrucciones 9, 10, 12, 13, 17 y 22 son instrucciones condicionales implícitas, de la forma «Ir a(condi-

ción)/Etiqueta», que significa: «Ir a Etiqueta si se cumple la condición». La flecha hacia la derecha representa la orden «Ir a».

6. Tampoco existen instrucciones explícitas de bucle. Para iniciar bucles basta con asignar un valor inicial a una variable cualquiera (como en las líneas 2 y 5). Para terminarlos, pondremos una instrucción condicional que haga avanzar la variable en cuestión, la compare con su valor final y, en caso de no haberlo rebasado, salte de nuevo al principio del bucle. (Véase, por ejemplo, la línea 22.) Pero hay otras formas de terminar el bucle, como la correspondiente a las líneas 17 a 19, que pone punto final al bucle interno (el de los cinco disparos).

7. La línea 0 contiene el nombre del programa. Bastará escribirlo (y presionar la tecla ENTER) para que éste se ejecute.

8. La línea 1 calcula el alcance del cañón como un número aleatorio comprendido entre 10.000 y 50.000. (La expresión ?40.000 genera un número aleatorio comprendido entre 1 y 40.000).

9. La función formato (⍕), que aparece en las líneas 3, 6, 14, 16 y 21, convierte el valor numérico situado a su derecha en la cadena de caracteres equivalente. Por ejemplo, el número 12 quedará convertido en la cadena de caracteres formada por un «1» seguido de un «2».

10. Cuando a una variable se le asigna un cuadrado, significa que el valor correspondiente se lee del teclado y es proporcionado por el jugador.

11. La línea 11 calcula la distancia alcanzada por la bala de cañón, y es idéntica a la línea 130 del programa BASIC, aunque con una notación diferente. El círculo a la izquierda de una expresión significa «multiplicarla por pi» (como en oE÷180). Cuando el círculo tiene un 1 a su izquierda, y una expresión a su derecha, significa que se desea calcular el seno de la expresión (como en 1o2×oE÷180).

Finalmente, el programa 2.3 presenta la realización de este juego en el lenguaje PASCAL.

Programa 2.3: Juego del disparo de cañón en lenguaje PASCAL.

Válido para máquinas que dispongan de un compilador de PASCAL (ver texto).

```
program DISPARO;

var
 a, d, e, r: real;
 i, j, n: integer;
 success: boolean;

begin
 a := 10000.+10.*random(4000);
 for n:=1 to 3 do
   begin
```

```
    writeln('El alcance de tu cañón es igual a ',a:5:0,' metros.');
    d := int(a/100*random(100));
    writeln('Hay un objetivo enemigo a una distancia de ',d:5:0,' metros.');
    i:=1;
    success:=false;
    while i<6 do
     begin
      write('Dame el ángulo de elevación de tu cañón, en grados: ');
      readln(e);
      if e<0.5 then
        writeln('El ángulo de elevación no puede ser menor que 0.5 grados.')
      else if e>89.5 then
        writeln('El ángulo de elevación no puede ser mayor que 89.5 grados.')
        else begin
          r := int(a*sin(2*3.1415927*e/180));
          if 100 >= abs(r-d) then begin
             j:=i;
             i:=6;
             success:=true;
             end
          else if r>d then writeln('Tu tiro se pasó en ',r-d:5:0,' metros.')
               else writeln('Tu tiro se quedó corto en ',d-r:5:0,' metros.');
        end;
      if success=false then writeln('Prueba otra vez.');
      i:=i+1
     end;

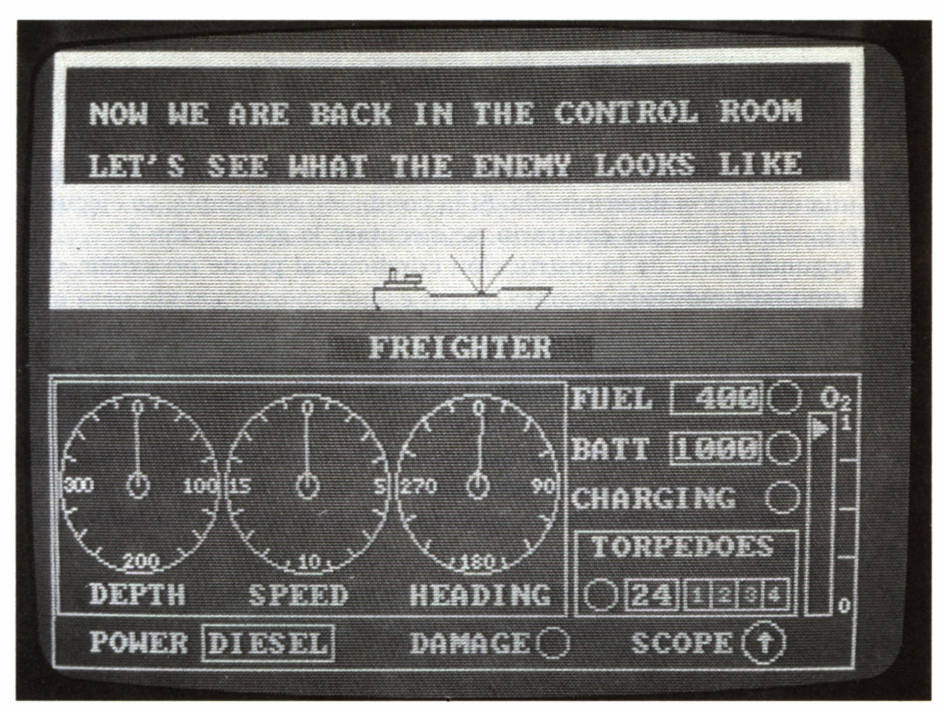
     if success then begin
        writeln('El objetivo enemigo saltó por los aires.');
        writeln('Lo has logrado en ',j,' intentos.');
        end
     else writeln('Lo siento. Tu unidad ha sido destruida por el enemigo.');
   end;
   writeln('Gracias por jugar conmigo.');
end.
```

Programa 2.3. *Juego del «Disparo de cañón», en lenguaje PASCAL.*

Las cuatro primeras líneas del programa 2.3 son declaraciones del tipo o formato interno de las variables que el programa va a utilizar. Algunas de estas variables serán «reales» (es decir, sus valores serán números decimales), otras son «enteras», y una de ellas (success) es «booleana», que tan sólo puede adquirir los valores «verdadero» y «falso» («true» y «false», en inglés).

El programa propiamente dicho comienza en la primera instrucción «begin».

Se observará que los programas en PASCAL se apoyan en una forma diferente de programar, que se llama «programación estructurada», que



La sala de control del juego «Gato» pone al alcance del jugador un sistema completo de información que simula los instrumentos de un submarino real e incluye información sobre el combustible disponible, los torpedos, velocidad, profundidad, etc.

prescinde de las instrucciones de transferencia incondicional de control (GOTO) y se basa en la utilización casi exclusiva de tan sólo tres construcciones fundamentales:

1. El «bloque de instrucciones», formado por dos o más instrucciones consecutivas encerradas entre las palabras reservadas «begin» y «end». Un bloque, en general, tiene la estructura siguiente:

```
begin
  instrucción-1;
  instrucción-2;
  ...
  instrucción-n;
end
```

2. La «instrucción condicional»:

```
if condición then instrucción-1 else instrucción-2
```

que selecciona cuál de las dos instrucciones debe ejecutarse, dependiendo de una condición determinada. Si la condición se cumple, se ejecutará la instrucción-1. En caso contrario, se ejecutará la instrucción-2.

La segunda parte de la instrucción condicional puede no existir, quedando entonces reducida a:

```
if condición then instrucción-1
```

que significa que se ejecutará la instrucción-1 si la condición se cumple, y no se ejecutará nada en caso contrario.

3. La «instrucción de bucle», que a su vez puede tener varias formas. Por ejemplo:

```
for variable:=valor-1 to valor-2 do instrucción
```

que indica que la instrucción será ejecutada varias veces sucesivamente. La variable indicada recibirá primero el valor-1, y cada vez que se ejecute la instrucción se incrementará su valor en una unidad, hasta que sobrepase el valor-2, en cuyo caso ya no se ejecuta más la instrucción correspondiente.

Otra forma de la instrucción de bucle es la siguiente:

```
while condición do instrucción
```

que indica que la instrucción será ejecutada varias veces sucesivamente, mientras se cumpla la condición especificada, y dejará de ejecutarse cuando dicha condición ya no se cumpla.

El programa de la figura 2.3 puede, por tanto, reducirse a un bloque que contiene las siguientes instrucciones:

```
begin
  a := 10000.+10.*random(4000);
  for n:=1 to 3 do bloque-2;
```

```
                    writeln('Gracias por jugar conmigo.');
                  end.
```

donde la primera instrucción genera el alcance del cañón (variable «a») como un número aleatorio comprendido entre 10.000 y 50.000. La expresión «random(4.000)» devuelve un valor entero aleatorio comprendido entre 1 y 4.000.

La segunda instrucción indica que el conjunto representado por «bloque-2» debe ejecutarse tres veces. Este bloque contiene una pasada completa del juego, desde que se presenta el objetivo a alcanzar hasta que éste es destruido o nuestro cañón ha fracasado en la empresa.

Por último, la tercera instrucción del bloque principal se limita a escribir un mensaje de despedida.

Veamos ahora, con mayor detalle, el bloque-2 que, a su vez, puede representarse (con notación de bloques) de la siguiente manera:

```
begin
  writeln('El alcance de tu cañón es igual a ',a:5:0,' metros.');
  d := int(a/100*random(100));
  writeln('Hay un objetivo enemigo a una distancia de ',d:5:0,' metros.');
  i:=1;
  success:=false;
  while i<6 do bloque-3;
  if success then bloque-4
     else writeln('Lo siento. Tu unidad ha sido destruida por el enemigo.');
end
```

La primera instrucción del bloque-2 escribe un mensaje que nos anuncia cuál es el alcance del cañón. El valor de «a» se escribe como un número de cinco cifras sin decimales (a:5:0).

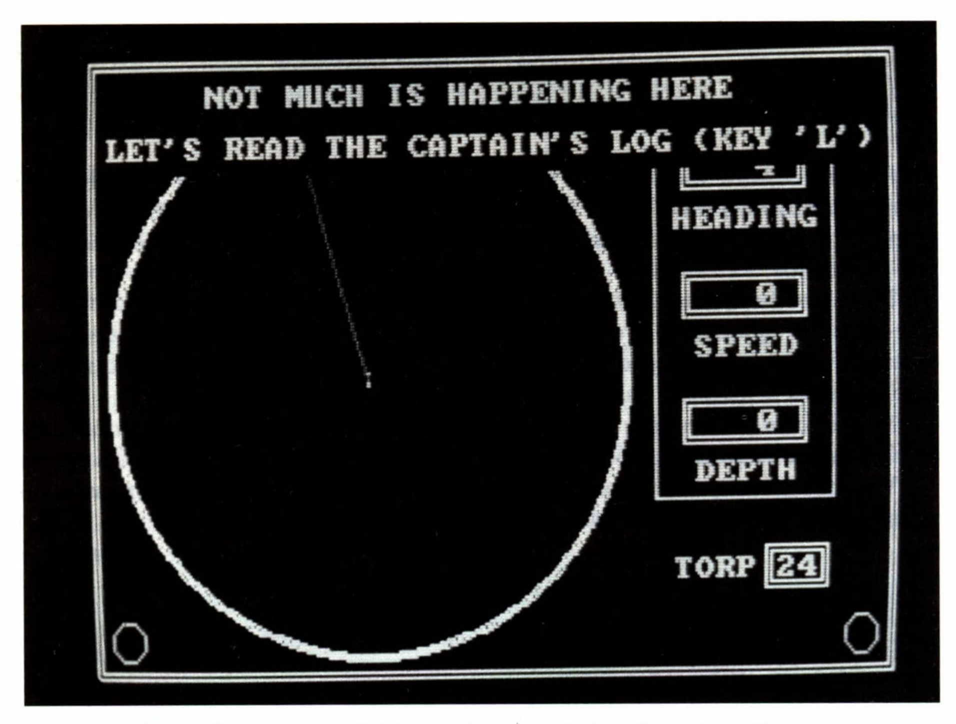
La segunda instrucción calcula la distancia a la que se encuentra el objetivo (variable «d») como un porcentaje aleatorio del alcance del cañón.

La tercera instrucción nos dice a qué distancia se encuentra el objetivo.

La cuarta instrucción inicializa el contador de disparos (variable «i»).

La quinta inicializa la variable booleana que indicará si nuestra acción ha tenido éxito o no (su valor inicial es «false»).

La sexta instrucción es un bucle que ejecuta el conjunto bloque-3. Este bloque contiene las instrucciones que corresponden a la realización de cada disparo.



Pantalla de sonar del juego de control de submarino «Gato».

Por último, la séptima instrucción escribe el resultado final de esta fase del juego. Si terminó con éxito, se ejecutan las instrucciones del bloque-4, que escriben un mensaje de felicitación, que nos dice además cuántos intentos nos ha costado destruir el objetivo. En caso contrario, se indica su fracaso con el mensaje correspondiente. El bloque-4 es muy sencillo, y se reduce a:

```
begin
  writeln('El objetivo enemigo saltó por los aires.');
  writeln('Lo has logrado en ',j,' intentos.');
end
```

Nos queda el bloque-3, la ejecución de un disparo. Se reduce a lo siguiente:

```
begin
  write('Dame el ángulo de elevación de tu cañón, en grados: ');
```

```
readln(e);
if e<0.5 then
 writeln('El ángulo de elevación no puede ser menor que 0.5 grados.')
 else if e>89.5 then
  writeln('El ángulo de elevación no puede ser mayor que 89.5 grados.')
  else bloque-5
if success=false then writeln('Prueba otra vez.');
i:=i+1
end
```

Este bloque contiene cinco instrucciones, una de las cuales (la tercera) es muy larga.

La primera instrucción escribe el mensaje que nos pide el ángulo de elevación del cañón.

La segunda obtiene el valor que escribimos en el teclado, y lo guarda en la variable «e».

La tercera es una instrucción condicional múltiple. Hemos visto antes que estas instrucciones tienen la forma:

```
if condición then instrucción-1 else instrucción-2
```

Ahora bien, tanto instrucción-1 como instrucción-2 pueden ser, a su vez, bloques, instrucciones condicionales o bucles. En este caso, instrucción-2 es una nueva instrucción condicional, por lo que el conjunto quedará de la forma:

```
if condición-1 then instrucción-1
   else if condición-2 then instrucción-3 else instrucción-4
```

Esta es, precisamente, la forma de la instrucción que nos ocupa:

```
if e<0.5 then
 writeln('El ángulo de elevación no puede ser menor que 0.5 grados.')
 else if e>89.5 then
  writeln('El ángulo de elevación no puede ser mayor que 89.5 grados.')
  else bloque-5
```

donde condición-1 es «e $<$ 0.5», instrucción-1 es

```
writeln('El ángulo de elevación no puede ser menor que 0.5 grados.')
```

condición-2 es «e > 89.5», instrucción-3 es

```
writeln('El ángulo de elevación no puede ser mayor que 89.5 grados.')
```

e instrucción-4 es «bloque-5», que analiza el disparo para ver si ha hecho blanco.

La cuarta instrucción del bloque-3, que estamos analizando, escribe el mensaje «Prueba otra vez» si nuestro disparo anterior no hizo blanco.

Por último, la quinta instrucción incrementa en una unidad el contador de intentos (la variable «i»).

Pasamos ahora al bloque-5 (el análisis del disparo), que puede escribirse así:

```
begin
  r := int(a*sin(2*3.1415927*e/180));
  if 100 >= abs(r-d) then bloque-6
   else if r>d then writeln('Tu tiro se pasó en ',r-d:5:0,' metros.')
         else writeln('Tu tiro se quedó corto én ',d-r:5:0,' metros.');
end
```

y que consta de dos instrucciones. La primera calcula la distancia alcanzada por el disparo («r»), en función del alcance del cañón («a») y del ángulo de elevación («»).

La segunda es también una instrucción condicional múltiple, del tipo:

```
if condición-1 then instrucción-1
   else if condición-2 then instrucción-3 else instrucción-4
```

donde condición-1 (100 > =abs(r-d)) comprueba si el disparo cayó a menos de 100 metros del objetivo, en cuyo caso hemos tenido éxito y se ejecuta instrucción-1 (el bloque-6). En caso contrario, ejecutamos una nueva instrucción condicional, cuya condición-2 (r > d) nos indica si el tiro se pasó o se quedó corto, escribiéndose en cada caso el mensaje correspondiente.

Por último, el bloque-6, que trata el caso de que nuestro disparo alcanzara el objetivo, se reduce a:

```
begin
  j:=i;
```

```
i:=6;
success:=true;
end
```

donde la primera instrucción guarda en la variable «j» el valor de «i» (para poder luego informarnos del número de intentos que tuvimos que realizar para destruir el objetivo), la segunda asigna al contador «i» el valor 6, para que se abandone inmediatamente del bloque «while», que corresponde a los diferentes intentos, y la tercera indica que hemos tenido éxito, asignando el valor «true» (verdad) a la variable booleana.

Los tres programas detallados en este capítulo son equivalentes en el sentido de que, en condiciones idénticas, producen los mismos resultados, aunque la intervención de números aleatorios, generados independientemente por los procesadores de cada lenguaje, pueden dar lugar a que los resultados concretos de las diferentes pasadas sean muy distintos entre sí. Sin embargo, con el mismo alcance del cañón y la misma distancia del objetivo, los tres programas darán a éste por alcanzado exactamente con el mismo ángulo.

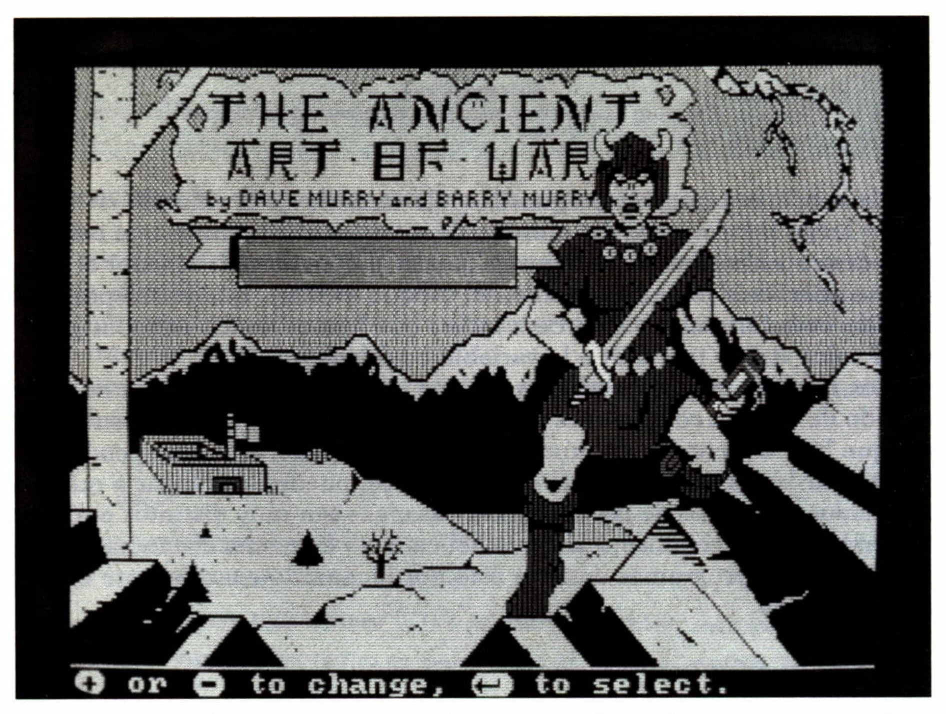


SIMULADORES DE COMBATE

Uno de los juegos más espectaculares de este tipo se llama «El Antiguo Arte de la Guerra», y ha sido desarrollado por Dave y Barry Murry y comercializado por Broderbund (TM) and Evryware (TM). El nombre del juego es la traducción del título de una obra china escrita hace más de dos milenios por el general Sun Tzu. Se trata de un simulador de combate que presenta sobre una pantalla gráfica situaciones supuestamente reales, donde el jugador toma el mando de una de las partes, mientras el programa dirige las fuerzas enemigas.

Los elementos fundamentales del «Antiguo Arte de la Guerra» son:

1. Una topografía que ocupa aproximadamente la superficie equivalente a tres pantallas, y que contiene los siguientes componentes: pueblos, fuertes, montañas, masas de agua (ríos, lagos o mares), bosques y caminos. El territorio donde va a desarrollarse la lucha puede ser una isla rodeada por el mar, una extensa llanura cruzada por un río, un valle entre montañas, un inmenso bosque con caminos o cualquier combinación de las componentes anteriores. Salpicados por el territorio aparecerán los signos de la actividad humana, representados por poblados y fuertes.
2. Una o varias banderas que hay que defender o capturar. Las banderas propias aparecen dibujadas en color blanco; las enemigas en negro.



«El antiguo arte de la guerra» es uno de los juegos de combate más espectaculares que existen. Los gráficos en color tienen una gran calidad y la variedad de los dibujos animados hace el juego singularmente atractivo.

Estas banderas pueden estar situadas en cualquier punto de la topografía, como en pueblos, fuertes, en las montañas o en campo abierto.

3. Un ejército propio, cuyas componentes aparecen en la pantalla en color blanco, y un ejército enemigo representado en color negro. Los miembros de estos ejércitos están agrupados en escuadras de cierto número de hombres, que pueden pertenecer a uno o varios de los cuatro grupos siguientes:

- Arqueros, que en caso de combate disparan flechas desde lejos.
- Caballeros, armados con espada, que combaten de cerca.
- Bárbaros, semejantes a los caballeros, pero que combaten con los puños y los pies.
- Espías, desarmados, rápidos y huidizos, que no combaten, pero pueden servir para localizar destacamentos enemigos ocultos.

Un ejército puede constar de una sola escuadra o de más de una docena. Una escuadra puede contener un solo hombre o un número cualquiera de ellos, hasta un máximo de catorce. Los miembros de una escuadra se desplazan juntos, aunque es posible separarlos en dos o más grupos, reu-



Los combates pre-programados en «El antiguo arte de la guerra» varían considerablemente de una a otra pasada del juego. En particular, la elección del enemigo influye en la estrategia del enemigo y en la marcha del combate.

nir varias escuadras o transferir hombres de una escuadra a otra, siempre que el número total de miembros de cada una no rebase los catorce. Además, los componentes de la escuadra pueden disponerse en diversas formaciones, que pueden elegirse entre varias preseleccionadas, de forma independiente para cada escuadra. Veamos algunos ejemplos:

1. Una escuadra de seis arqueros (representados por la letra A), situados en línea en el extremo derecho de la pantalla, se representa así:

A

A

A

A

A

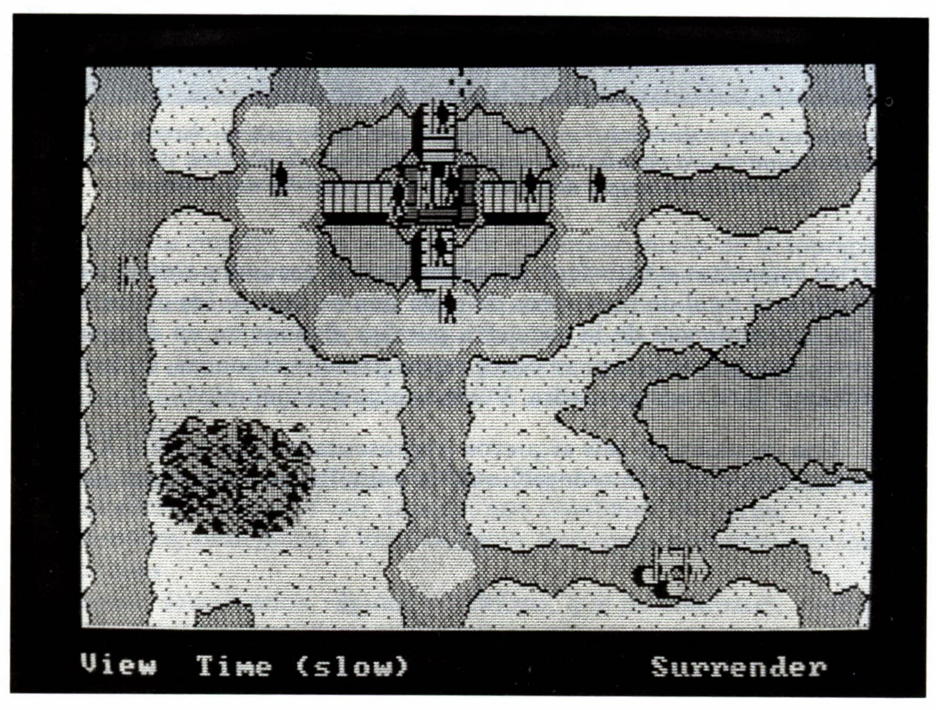
A

2. Una escuadra de ocho bárbaros (B) y seis caballeros (K) podría adoptar la siguiente formación:

B B B
K B
K K
K K
K B
B B B

3. Una escuadra de dos arqueros, cuatro bárbaros y cuatro caballeros, podría disponerse de la siguiente forma:

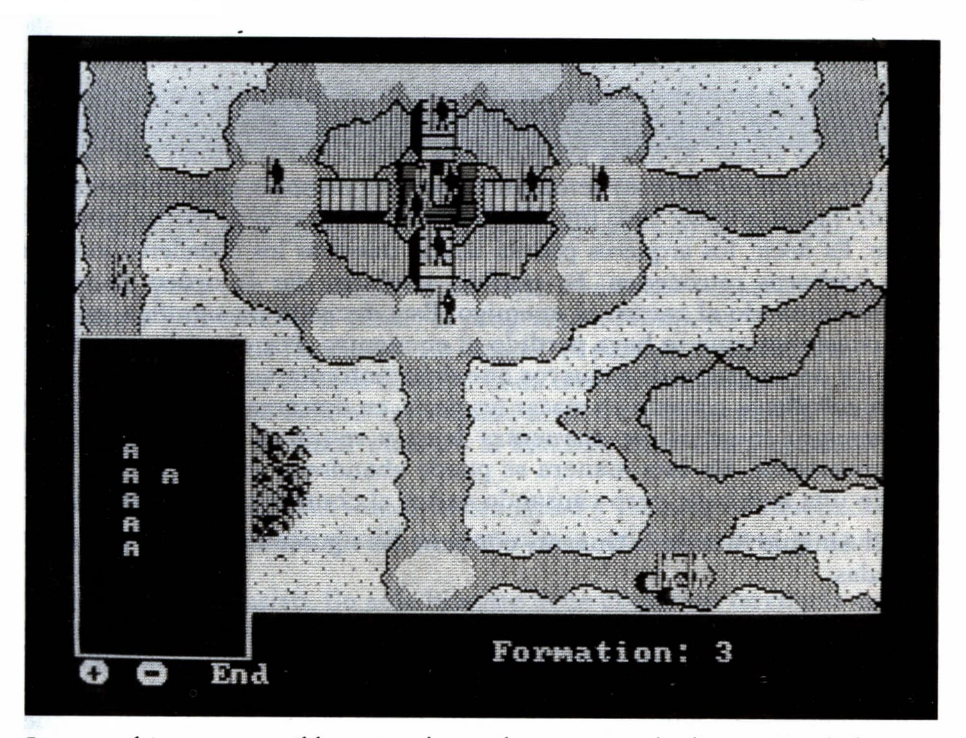
A
B K
B K
B K
B K
A



En todo momento, el jugador tiene ante sus ojos un mapa que le presenta la topografía del terreno de combate. Los distintos elementos del terreno (poblados, fuertes, montañas, etc.), están dibujados de modo esquemático pero realista.

Naturalmente, la disposición de las tropas en cada escuadra afecta a los resultados de los combates en que participe. El jugador puede modificar la composición de las escuadras (dividiéndolas, uniéndolas o transfiriendo guerreros) y puede seleccionar la formación que le parezca más adecuada para el tipo de combate en que habrán de tomar parte.

Cada escuadra puede también moverse independientemente a través de la topografía del terreno de lucha. La velocidad de la marcha puede graduarse y hacerse lenta, normal o rápida. No todas las formas de marcha son siempre posibles, sin embargo, pues cada escuadra está provista de cierta cantidad de alimentos (que va consumiendo a lo largo del tiempo, pero puede reponer en lugares predeterminados) y se encuentra en cada momento en cierto estado o forma física (desde un agotamiento máximo, pasando por un cansancio relativo, hasta una forma óptima, que puede adquirir descansando). Una escuadra cansada no conseguirá realizar una marcha rápida. Una escuadra casi agotada sólo podrá moverse a marcha muy lenta. Una escuadra totalmente agotada deberá permanecer parada y no podrá comportarse adecuadamente en combate contra el enemigo. Por



Durante el juego, es posible variar de muchas maneras la disposición de los componentes de cada uno de los grupos de guerreros que están bajo el control del jugador. En la fotografía se puede observar la disposición de un grupo de seis arqueros, representados con la letra A.

último, una escuadra puede perecer por hambre o ser capturada por el enemigo por desigualdad manifiesta de fuerzas.

La velocidad de movimiento de las tropas y la rapidez de su cansancio puede verse afectada, también, por el tipo de terreno que están atravesando. Por ejemplo, al cruzar un río, el agua puede ser profunda y rápida, profunda y tranquila, o muy fácil de cruzar. Si una escuadra alcanza el agotamiento mientras está cruzando aguas profundas, se ahogará. Lo denso o espeso del bosque, la altura de las montañas, son otras circunstancias que afectan también la marcha de los ejércitos.

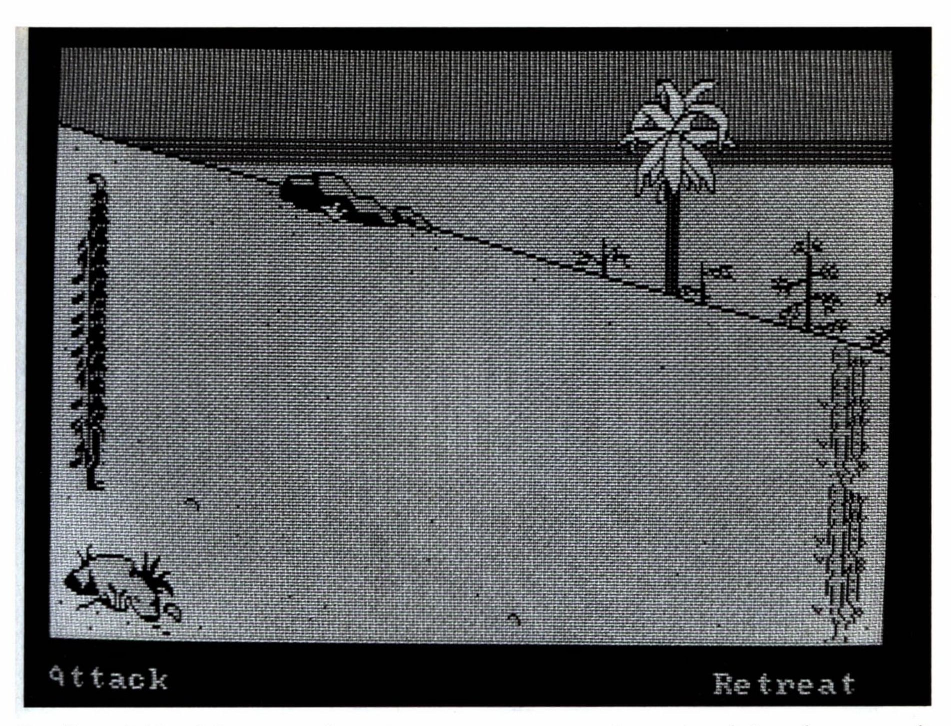
Cuando una escuadra enemiga está situada muy cerca de una propia, se produce un encuentro. Entonces el programa permite realizar un «zoom» (es decir, un acercamiento de la imagen) para ver y controlar de cerca el combate, con una velocidad de avance del tiempo muy realista. La imagen que aparece en la pantalla en caso de encuentro varía según las circunstancias: se podrá ver un paisaje diferente según se trate de un combate en zona montañosa, en las orillas de un río, en un poblado, en campo abierto, en el asalto a un fuerte, etc. Además, los miembros de las dos escuadras en conflicto (la propia y la enemiga) aparecerán dibujados en la pantalla en la formación correspondiente.

Durante el encuentro propiamente dicho, el jugador puede controlar las acciones de los miembros de su escuadra, haciéndolos adelantarse, retroceder o atacar. Si todos los guerreros pertenecen al mismo tipo (arqueros, bárbaros o caballeros) actuarán siempre al unísono, pero si hay miembros de varios grupos, cada uno de ellos podrá moverse de forma independiente. Es decir, los arqueros podrán atacar mientras los caballeros avanzan y los bárbaros retroceden. Es posible, también, dar la orden de retirada, abandonando el campo al enemigo. Esta decisión no significa, en muchas ocasiones, que se haya perdido el encuentro. Por ejemplo, una táctica típica de un grupo de arqueros que debe enfrentarse contra una escuadra de caballeros consiste en atacar (es decir, en descargar sus flechas contra el enemigo) mientras se encuentra lejos y retirarse cuando se acerca peligrosamente. De esta manera se puede abandonar el campo sin haber sufrido una sola baja, pero causando varias al contrario y debilitándole.

«El Antiguo Arte de la Guerra» contiene las siguientes batallas preseleccionadas:

1. La carrera por las banderas. Se trata de una situación supuestamente relacionada con la invasión mongol de China. El ejército enemigo nos intercepta el paso para llegar hasta un lugar relativamente alejado donde se encuentran dos banderas: una propia y otra contraria. El primero que llegue hasta ellas ganará la partida.

2. La batalla de Farsalia, que en realidad tuvo lugar entre César y Pompeyo en el año 48 antes de Cristo. César tenía muchas menos tropas que



La disposición del terreno durante un encuentro varía según el tipo de topografía donde éste tenga lugar. La disposición de los guerreros propios está siempre de acuerdo con la elegida por el jugador para cada uno de sus grupos.

Pompeyo, pero con sus hábiles maniobras consiguió vencerle. El jugador representa en este caso a César, el programa a Pompeyo.

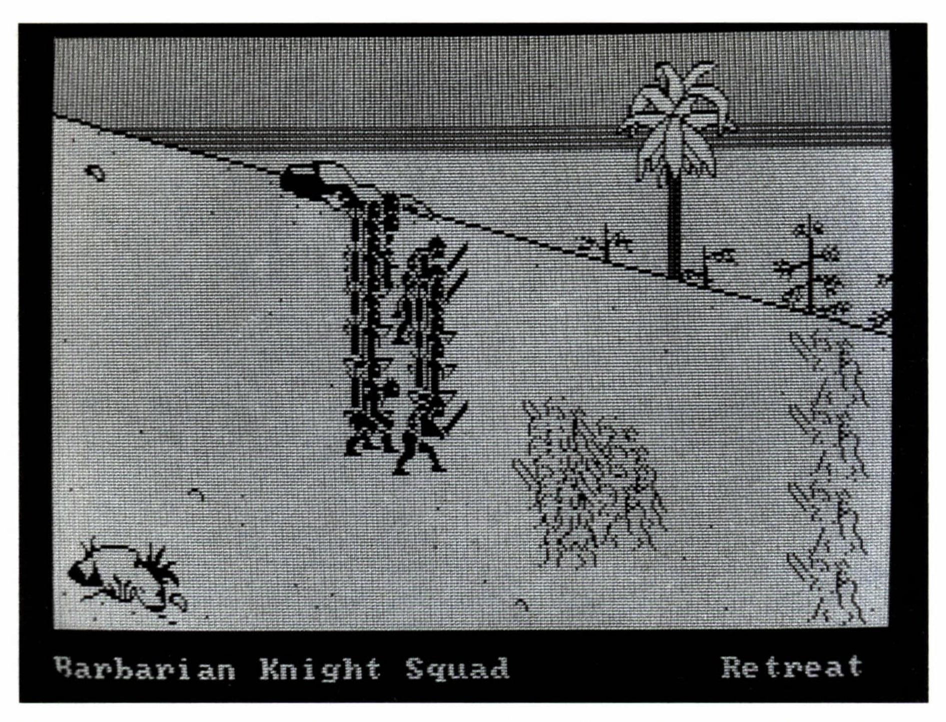
3. La disputa de los dioses. En Asgard, el paraíso nórdico, Thor y Loki se enfrentan en una curiosa disputa. Cada uno de ellos elige un campeón, un solo hombre que deberá enfrentarse al otro en un torneo de inteligencia y rapidez para alcanzar primero la bandera contraria y apoderarse de ella.

4. El bosque de Sherwood. Robin Hood y sus alegres muchachos luchan contra el malvado sheriff de Nottingham en los espesos bosques que rodean su castillo.

5. El espía elusivo. Un solo hombre debe cruzar un territorio extenso y plagado de enemigos, para apoderarse de su bandera sin ser capturado en el intento.

6. La última defensa de Custer. La famosa batalla de Little Big Horn, donde el ejército del general Custer (representado por el jugador) fue totalmente destruido por la confederación de los indios Sioux, bajo el mando de Sitting Bull (Toro Sentado).

7. La rivalidad. Un rey decide poner a prueba a sus dos hijos para des-



El combate propiamente dicho está parcialmente bajo el control del jugador, que puede elegir independientemente los movimientos a realizar por sus bárbaros, arqueros y caballeros. También le queda la posibilidad de ordenar la retirada.

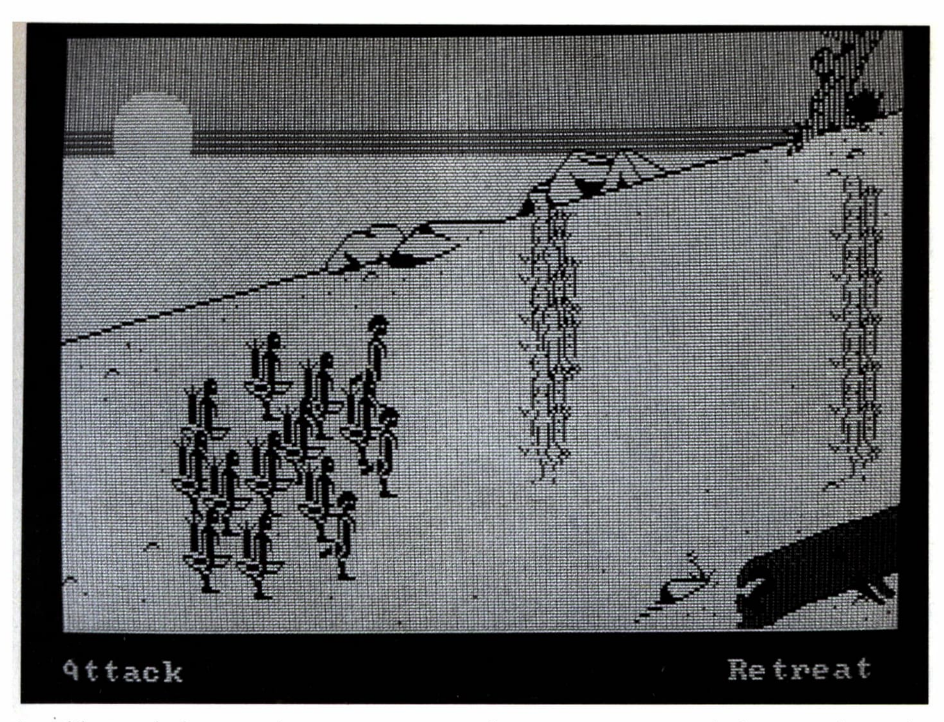
cubrir quién es más digno de heredar el reino. Para ello, entrega a cada uno una isla donde hay un fuerte y una bandera, así como un pequeño ejército. El que consiga conquistar al otro será declarado vencedor. Desgraciadamente para el jugador, el otro hijo es el preferido, y el padre le ha concedido más tropas...

8. El relato de las tres islas. Durante las campañas de Napoleón, la isla del norte ha sido ocupada por el Gran Corso. El jugador debe impedirle que conquiste también la isla del sur, donde se encuentra con sus tropas, y vencerle si es posible.

9. Wu frente a Ch'u. Dos reinos chinos se enfrentan, 400 años antes de Cristo, en una guerra sin cuartel. El jugador debe defender el reino de Ch'u contra la conquista y la desaparición total, amenazada por el general enemigo Sun Tzu, autor de «El Antiguo Arte de la Guerra».

10. Guerra en las montañas. Las tropas norteamericanas persiguen al gran jefe apache Gerónimo, que ha salido sin permiso del territorio indio para buscar nuevos campos de caza.

11. Las islas del destino. De nuevo nos encontramos en el campo de



Los dibujos de la zona de encuentro entre los guerreros controlados por el jugador (representados en blanco) y los del enemigo, controlados por el programa (que aparecen en negro) son extraordinariamente detallados.

acción de la invasión mongol de China. El ejército enemigo ha tomado las islas, donde sólo queda un puñado de valientes defensores.

Una vez que el jugador ha elegido la batalla en la que quiere tomar parte, se le proporciona una breve información histórica respecto a la misma y se le dan a elegir las condiciones del juego, a saber:

- Si en los poblados y fuertes se encuentra comida o no.
- Si las líneas de abastecimiento son largas, medianas o cortas.
- Si los hombres propios comienzan en forma excelente, buena o mala.
- Si al enemigo se le ve siempre, de lejos, o sólo de cerca.
- Si los cursos de agua son fáciles de cruzar, profundos y tranquilos o profundos y rápidos.
- Si los territorios montañosos son altos y arriesgados, altos y seguros o bajos y seguros.
- Si los bosques son abiertos o densos.

A continuación, el jugador debe elegir su contrincante entre ocho posibles: la diosa Atenea, Alejandro Magno, Gerónimo el apache, el zar Iván



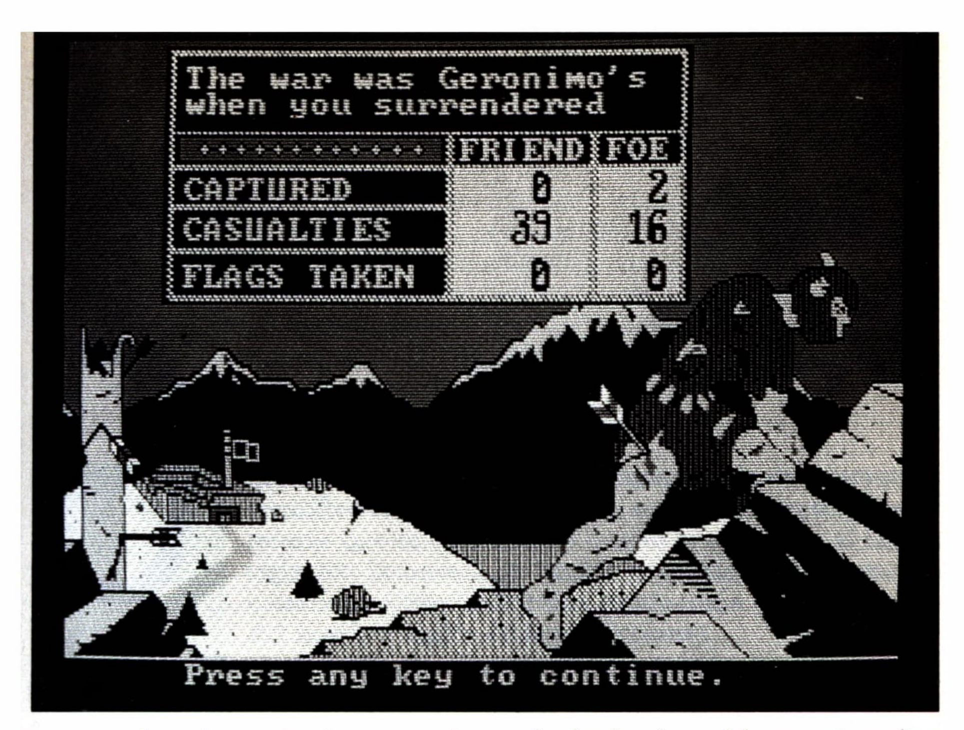
Durante el ataque se producen encuentros individudles entre los diversos miembros de los dos grupos, vuelan las flechas y se presentan efectos de animación muy complejos.

el Loco, Julio César, Gengis Khan, Napoleón o Sun Tzu. Cada uno de estos jefes tiene sus puntos fuertes y sus debilidades, por lo que la actuación del enemigo dependerá del jefe seleccionado.

A partir de este momento comienza la lucha. El juego termina cuando ocurre una de las circunstancias siguientes:

1. Todas las banderas de uno de los ejércitos han sido capturadas por el otro.
2. Todas las escuadras de uno de los ejércitos han sido destruidas o capturadas por el otro.
3. Todas las escuadras de uno de los ejércitos se encuentran totalmente agotadas.
4. Uno de los contrincantes ofrece rendirse, y el otro lo acepta. Si es el jugador el que se rinde, su ofrecimiento será siempre aceptado por el programa. Si sucede al contrario, el jugador puede rechazar la rendición y exigir al programa que continúe la lucha hasta la destrucción total.

Una vez terminado el combate, se ofrece al jugador una estadística de los resultados finales: quién ha vencido, cuántas banderas ha conquistado



Vence en el combate quien logra conquistar todas las banderas del contrario o eliminar o capturar a todos sus hombres. También es posible que uno de los lados decida rendirse.

cada grupo, cuántos hombres han caído en cada ejército y cuántos fueron capturados.

Por último, el programa posee una función adicional que abre numerosas posibilidades: la creación de situaciones nuevas, a gusto del jugador. Cuando se utiliza esta alternativa, se puede seleccionar la topografía deseada, situar poblados y fuertes en el mapa, colocar banderas y escuadras propias y contrarias, decidir las fuerzas y disposición de cada una de ellas, etcétera. De este modo, el juego seguirá siendo interesante y presentará nuevos desafíos, incluso en el caso de que el jugador haya logrado dominar las situaciones prefabricadas, lo que no es fácil, ni mucho menos.

AVENTURAS ESPACIALES 3



URANTE el siglo XIX, como consecuencia de la revolución científico-técnica de la época, surgió un nuevo género literario que se consolidó y creció con rapidez al llegar el siglo XX: la novela de ficción científica, mal llamada en castellano «ciencia-ficción» (traducción demasiado literal del nombre inglés «science fiction»).

Al principio, durante el siglo pasado, las incursiones en el campo de la novela de anticipación (como también se llama este género) vinieron de la mano de las grandes plumas de la literatura: Ernst Hoffmann, Mary Shelley, Edgar Allan Poe, Guy de Maupassant, Nathaniel Hawthorne... Pero, ya avanzado dicho siglo, tomaron la antorcha dos escritores a los que se considera corrientemente como los padres de la literatura de ficción científica, como consecuencia del gran número de sus obras en este campo y del alto grado de popularidad de que gozaron, aun cuando ninguno de ellos fue realmente su iniciador, y ambos dedicaron gran parte de sus esfuerzos a otros géneros literarios distintos. Me refiero al francés Jules Verne (1828-1905) y el inglés Herbert George Wells (1866-1946).

A partir de principios del siglo XX, la influencia de estos dos escritores provocó una proliferación impresionante de la ficción científica, pero este fenómeno coincidió con una considerable reducción del valor estético de las producciones, lo que desacreditó el género y lo redujo a un nivel subliterario del que lucharon infructuosamente por sacarlo escritores de nivel medio como Poul Anderson, Isaac Asimov, Ray Bradbury, Arthur C. Clarke, C. S. Lewis y Walter M. Miller Jr. En la actualidad, la ficción científica es despreciada por muchos grandes escritores y por la práctica totalidad de los críticos, aunque eso no impide que el género haya sido abordado ocasionalmente por plumas tan notorias como las de André Maurois, William Golding, Aldous Huxley y algunos más.

La literatura de anticipación incluye varios temas principales: el pri-

mero trata de describir cómo podría ser la sociedad del futuro, qué efectos puede tener el desarrollo técnico y científico sobre la misma, cuál será su evolución moral o política. En esta rama es necesario citar *Un mundo feliz,* de Aldous Huxley, *Mil novecientos ochenta y cuatro,* de George Orwell, *La máquina de leer los pensamientos,* de André Maurois, la famosa trilogía de la *Fundación,* las historias de robots y *Las cavernas de acero,* de Isaac Asimov, y *Esa fuerza horrible,* de C. S. Lewis.

En un segundo grupo podríamos incluir todas las novelas que describen los efectos de una guerra nuclear sobre los habitantes de la Tierra y la estructura de la sociedad: *Un cántico a San Leibowitz,* de Walter M. Miller Jr., es su exponente más notable. *El señor de las moscas,* de William Golding, podría también clasificarse en este apartado, aunque en este caso la guerra nuclear proporciona sólo un pretexto no imprescindible para el desarrollo del argumento.

El tercer grupo de esta clasificación no exhaustiva de temas comprende las novelas que presentan situaciones relacionadas con los viajes en el tiempo, ya sea hacia el pasado o hacia el futuro. Podemos citar aquí *La máquina del tiempo,* de H. G. Wells, *Los guardianes del tiempo,* de Poul Anderson, *La zorra y el bosque,* de Ray Bradbury, y varias de las obras del astrónomo inglés Fred Hoyle.

Por último, el tema más importante de la literatura de ficción científica (al menos en lo que respecta al número de obras producidas) tiene que ver con la exploración del espacio y el encuentro del hombre con civilizaciones extraterrestres. Nombraré aquí *De la Tierra a la Luna* y *Alrededor de la Luna,* de Jules Verne, *La guerra de los mundos* y *Los primeros hombres en la Luna,* de H. G. Wells, *Orbita ilimitada,* de Poul Anderson, *Las crónicas marcianas,* de Ray Bradbury, *2001, una odisea del espacio,* de Arthur C. Clarke, *La nube negra,* de Fred Hoyle, y *Perelandra,* de C. S. Lewis.

Ha sido precisamente este último grupo de temas (las aventuras espaciales) el que se ha convertido en el paradigma de todo el género y el que ha dado lugar a la aparición de numerosas adaptaciones al cine y a la televisión, algunas de dudoso valor, pero que han sido, y siguen siendo, extraordinariamente populares. Destacan, por su calidad técnica y por el número de recursos utilizados, la película de Stanley Kubrick *2001, una odisea del espacio, La guerra de las galaxias,* de Georges Lucas, y la serie de televisión *Star trek (El viaje estelar),* que se ha mantenido durante años en los más altos lugares de audiencia.

Con una popularidad semejante, la aventura espacial no podía dejar de ser una de las primeras en incorporarse al nuevo mundo de los juegos de ordenador. Efectivamente, así ha sido. Basta mirar cualquier catálogo de juegos para convencerse de que la categoría del «espacio» es una de las más importantes y numerosas.

Existen dos clases principales de aventuras espaciales: las de exploración (la más famosa es el «Desembarco lunar») y las de guerra espacial. Es-

tas últimas son en realidad casos particulares de los juegos descritos en el capítulo anterior, pero creo que es conveniente considerarlas por separado, dado su carácter de ficción científica.



UN JUEGO DE DESEMBARCO LUNAR

Vamos a ver, como ejemplo, el juego del desembarco lunar. En el momento en que comienza el juego se supone que el jugador se encuentra situado en una cápsula espacial a cierta distancia (entre 100 y 200 kilómetros) de la superficie de la Luna y que desea desembarcar en ésta. El programa conoce el peso de la cápsula y, por tanto, puede calcular los efectos sobre la misma de la gravedad lunar, que no debe olvidarse que es seis veces menor que la terrestre y, por ello, igual a 1,6 metros por segundo al cuadrado. (No tendremos en cuenta, en este programa, que la gravedad desciende con la altura respecto a la superficie de la Luna.)

El jugador conoce en cada momento la cantidad de combustible de que dispone y puede decidir qué cantidad de éste hace pasar a los cohetes de frenado. De este modo puede controlar la velocidad de su vehículo mientras va descendiendo. El objetivo del juego es tomar tierra en la superficie de la Luna sin estrellarse. Es decir, en el instante en que la altura se reduce a cero, la velocidad de la nave debe ser también muy próxima a cero (o, dicho de otro modo, la nave debe estar prácticamente parada en el momento del alunizaje).

En la versión que presento aquí, el peso de la cápsula y del combustible han sido elegidos de tal manera que se aproximan a los que realmente tenían las misiones Apolo que pusieron pie en la Luna hace poco más de una década.

Programa 3.1: Juego del desembarco lunar en lenguaje BASIC.

Este programa es válido para SPECTRUM, AMSTRAD, COMMODORE e IBM PC o compatibles (ver notas).

```
10 REM Desembarco lunar. Versión de M. Alfonseca
20 RANDOMIZE
30 LET A=100000+INT(RND*100000): REM Altura inicial en metros
40 LET PE=15000: REM Peso de la cápsula en kilos
50 LET CO=7500: REM Peso del combustible disponible, en kilos
60 LET V=0.625: REM Velocidad en km/seg
70 LET G=0.0016: REM Aceleración de la gravedad en la luna, km/s2
80 LET Z=1.8
90 PRINT "Desembarco lunar"
```

```
100 PRINT "Te encuentras a bordo de una cápsula lunar que está a punto
de"
110 PRINT "alunizar. En cada momento debes escoger la cantidad de"
120 PRINT "combustible que vas a dirigir a los cohetes de frenado."
130 PRINT "Puedes escoger entre 0 y 200 kg/seg cada 10 segundos."
140 PRINT "Elige con cuidado, para no estrellarte."
150 PRINT "Comienza el juego.": PRINT
160 PRINT "T=TIEMPO A=ALTURA V=VELOCIDAD"
170 PRINT "C=COMBUSTIBLE FRN=FRENADO": PRINT
180 PRINT "T (s) A (km)  V (km/h) C (kg) FRN"
190 PRINT SE;TAB(7);INT(A);TAB(15);0.1*INT(36000*V);TAB(24);CO;
200 INPUT FR
210 IF FR>200 THEN PRINT "Frenado inválido. Repite.": GOTO 190
220 IF FR<0 THEN PRINT "Frenado inválido. Repite.": GOTO 190
230 LET T=10: REM Tiempo hasta la próxima petición
240 IF CO<0.001 THEN GOTO 1000
250 IF T<0.001 THEN GOTO 190
260 LET S=T: REM S es el número de segs que dura el combustible a este
ritmo
270 IF CO>=S*FR THEN GOTO 300
280 LET S=CO/FR
300 GOSUB 2000
310 IF A1<=0 THEN GOTO 1200
320 IF V<=0 THEN GOTO 340
330 IF A2<0 THEN GOTO 1400
340 GOSUB 3000
350 GOTO 240
1000 PRINT "Se acabó el combustible después de ";SE;"segundos."
1010 LET S=(SQR(V*V+.002*A*G)-V)/G
1020 LET V=V+G*S: REM Velocidad final en km/seg
1030 LET SE=SE+S
1040 LET V1=3600*V: REM Velocidad final en km/h
1050 PRINT "Alcanzada la superficie de la Luna en";SE;"segundos."
1060 PRINT "Velocidad de impacto = ";INT(V1);"km/h."
1070 IF V1<=2 THEN PRINT "Aterrizaje perfecto. Enhorabuena.": GOTO
1110
1080 IF V1<=20 THEN PRINT "Has causado daños a la cápsula.": GOTO
1110
1090 PRINT "Te has estrellado contra la Luna. No hubo supervivientes."
1100 PRINT "Has abierto un cráter de";INT(V1*.052);"metros."
1110 STOP
1200 IF S<.005 THEN GOTO 1040
1210 LET D=V+SQR(V*V+.002*A*(G-Z*FR/PE))
1220 LET S=.002*A/D
1230 GOSUB 2000
1240 GOSUB 3000
1250 GOTO 1200
1400 LET V1=(1-PE*G/(Z*FR))/2
1410 LET S=PE*V/(Z*FR*(V1+SQR(V1*V1+V/Z)))+.05
1420 GOSUB 2000
```

```
1430 IF A1<=0 THEN GOTO 1200
1440 GOSUB 3000
1450 IF A2>0 THEN GOTO 240
1460 IF V>0 THEN GOTO 1400
1470 GOTO 240
2000 LET Q=S*FR/PE
2010 LET A2=V+G*S+Z*(-Q*(1+Q*(.5+Q*(.333333+Q*(.25+Q/5)))))
2020 LET A1=A-1000*(G*S*S/2+V*S-Z*S*Q*(.5+Q*(.1666667+Q*(1/12+Q*
(.05+Q/30)))))
2030 RETURN
3000 LET SE=SE+S
3010 LET T=T-S
3020 LET PE=PE-S*FR
3030 LET CO=CO-S*FR
3040 LET A=A1
3050 LET V=A2
3060 RETURN

NOTAS: El programa anterior es válido directamente para SPECTRUM.

Para AMSTRAD cambiar la siguiente línea:

 30 LET A=100000+INT(RND(1)*100000): REM Altura inicial en metros

Para COMMODORE cambiar las siguientes líneas:

 20 LET X=RND(-TI)
 30 LET A=100000+INT(RND(1)*100000): REM Altura inicial en metros

Para IBM PC cambiar la siguiente línea:

 1110 END
```

Programa 3.1. *Juego del «Desembarco lunar» en lenguaje BASIC.*

Las instrucciones números 20 y 30 del programa 3.1 permiten que las distintas pasadas del juego se diferencien en la altura inicial, que será un número aleatorio comprendido entre 100.000 y 200.000 metros. Si se desea, estas dos instrucciones pueden sustituirse por la asignación de un valor determinado a la variable ALT (por ejemplo, ALT=150000!).

Veamos ahora algunos ejemplos de su funcionamiento. En el primero, el control del combustible es muy exacto, y la cápsula llega a su destino en perfectas condiciones. Obsérvese que el jugador ha dejado la cápsula en situación de caída libre durante los dos primeros minutos, con lo que la distancia a la superficie ha disminuido muy deprisa (de más de 154 a algo menos de 60 kilómetros) sin gasto alguno de combustible, mientras

que la velocidad se ha acelerado desde 2.250 hasta casi 3.000 kilómetros por hora. En ese instante, sin embargo, el comandante de la cápsula comienza a aplicar combustible a los cohetes de frenado, hasta alcanzar un máximo de 80 kilogramos por segundo hacia los tres minutos de vuelo. Para entonces, la altura ha descendido a sólo 11 kilómetros y la velocidad se ha reducido hasta los 1.500 kilómetros por hora. A partir de este momento comienza una etapa de deceleración más pausada, que lleva a la cápsula a una altura de unos 150 metros y una velocidad de menos de 10 kilómetros por hora tras cinco minutos de vuelo. Por último, durante la cuarta fase, la aproximación final, se trata de mantener la velocidad lo más baja posible con ajustes muy finos, mientras la nave recorre los últimos metros del descenso. La duración total del vuelo ha sido inferior a los seis minutos y medio.

Random number seed (-32768 to 32767)? 0
Desembarco lunar
Te encuentras a bordo de una cápsula lunar que está a punto de
alunizar. En cada momento debes escoger la cantidad de
combustible que vas a dirigir a los cohetes de frenado.
Puedes escoger entre 0 y 200 kg/seg cada 10 segundos.
Elige con cuidado, para no estrellarte.
Comienza el juego.

Tiempo (s)	Altura (km)	Velocidad (km/h)	Combustible (kg)	Frenado?
0	154640	2250	7500	? 0
10	148310	2307.6	7500	? 0
20	141820	2365.2	7500	? 0
30	135170	2422.8	7500	? 0
40	128360	2480.4	7500	? 0
50	121390	2538	7500	? 0
60	114260	2595.599	7500	? 0
70	106970	2653.199	7500	? 0
80	99520	2710.799	7500	? 0
90	91910	2768.399	7500	? 0
100	84140	2825.999	7500	? 0
110	76210	2883.599	7500	? 0
120	68120	2941.199	7500	? 0
130	59870	2998.799	7500	? 20
140	51580	2969.418	7300	? 40
150	43497	2849.472	6900	? 40
160	35754	2724.525	6500	? 60
170	28498	2498.284	5900	? 60
180	21887	2259.038	5300	? 80
190	16106	1898.428	4500	? 80
200	11367	1508.954	3700	? 80
210	7754	1086.335	2900	? 70
220	5276	692.409	2200	? 60
230	3841	336.2515	1600	? 10
240	2927	322.2486	1500	? 10

250	2052	307.4456	1400	? 10
260	1219	291.8244	1300	? 20
270	535	200.4523	1100	? 30
280	216	27.96898	800	? 10
290	167	7.02256	700	? 7
300	144	9.068022	630	? 8
310	128	2.588392	550	? 5
320	97	19.81442	500	? 8
330	52	12.28824	420	? 8
340	29	4.100863	340	? 7
350	19	3.583874	270	? 7
360	10	2.54093	200	? 7
370	5	.9624302	130	? 6.7
380	2	1.40942	63	? 6.6

Alcanzada la superficie de la Luna en 384.446 segundos.
Velocidad de impacto =1 km/h.
Aterrizaje perfecto. Enhorabuena.

En el segundo ejemplo las cosas suceden de una manera muy diferente. En este caso, el capitán comienza el frenado demasiado pronto, cuando la nave se encuentra a más de 150 kilómetros de altura, y lo realiza demasiado bruscamente y con impulsos de combustible muy desiguales. En consecuencia, a los 260 segundos de vuelo, la cápsula, que todavía está a más de 100 kilómetros de la Luna, ha frenado hasta tal punto que comienza a alejarse de ella (la velocidad de aproximación se hace negativa). El capitán se pone nervioso y deja la nave en caída libre, pero la situación vuelve a repetirse en el segundo 300. En fin, el gasto de combustible ha sido tan exagerado, que éste se agota cuando la cápsula se encuentra todavía a más de 100 kilómetros de altura. Puede imaginarse el resultado: tal como nos dice el programa, la nave se ha estrellado contra la Luna a una velocidad de más de 2000 kilómetros por hora, abriendo un cráter de 107 metros de profundidad y pereciendo todos sus ocupantes.

Random number seed (-32768 to 32767)? 567
Desembarco lunar
Te encuentras a bordo de una cápsula lunar que está a punto de
alunizar. En cada momento debes escoger la cantidad de
combustible que vas a dirigir a los cohetes de frenado.
Puedes escoger entre 0 y 200 kg/seg cada 10 segundos.
Elige con cuidado, para no estrellarte.
Comienza el juego.

Tiempo (s)	Altura (km)	Velocidad (km/h)	Combustible (kg)	Frenado?
0	196802	2250	7500	? 0
10	190472	2307.6	7500	? 0
20	183982	2365.2	7500	? 0
30	177332	2422.8	7500	? 0

40	170522	2480.4	7500	? 0
50	163552	2538	7500	? 100
60	157035	2148.526	6500	? 10
70	151052	2159.674	6400	? 100
80	145636	1733.467	5400	? 10
90	140811	1740.639	5300	? 10
100	135966	1747.415	5200	? 100
110	131760	1273.57	4200	? 100
120	128935	752.2143	3200	? 10
130	126850	748.9688	3100	? 10
140	124774	745.1466	3000	? 10
150	122711	740.7365	2900	? 10
160	120660	735.7273	2800	? 10
170	118624	730.1073	2700	? 10
180	116604	723.8644	2600	? 10
190	114603	716.9862	2500	? 10
200	112622	709.4601	2400	? 20
210	110754	634.8106	2200	? 20
220	109097	557.4058	2000	? 20
230	107660	477.1282	1800	? 20
240	106449	393.8531	1600	? 20
250	105475	307.4472	1400	? 50
260	105056	−9.623651	899.9999	? 0
270	105003	47.97635	899.9999	? 10
280	104897	27.97061	800	? 10
290	104848	7.024183	700	? 10
300	104859	−14.88603	600	? 0
310	104820	42.71396	600	? 0
320	104622	100.314	600	? 0
330	104263	157.914	600	? 10
340	103856	135.016	500	? 10
350	103514	111.1055	400	? 10
360	103240	86.15657	300	? 10
370	103036	60.14248	200	? 10
380	102906	33.03537	100	? 10

Se acabó el combustible después de 390 segundos.
Alcanzada la superficie de la Luna en 747.7299 segundos.
Velocidad de impacto =2065 km/h.
Te has estrellado contra la Luna. No hubo supervivientes.
Has abierto un cráter de 107 metros.



UN JUEGO DE GUERRA ESPACIAL

Vamos a ver ahora un juego de guerra espacial, desarrollado por el autor de este libro a partir de una versión anterior original de M. Mayfield. Las instrucciones del juego son las siguientes:

Como capitán de la nave espacial «Aventura», el jugador debe cumplir la misión de localizar y destruir una flota de naves enemigas que han in-

vadido la Galaxia para destruir la Federación de Planetas Unidos. La misión debe llevarse a cabo en un número determinado de fechas estelares. Existen también bases estelares donde el Aventura puede abastecerse.

La galaxia se divide en 64 cuadrantes, dispuestos en forma de tablero de ajedrez de 8 filas y 8 columnas. Cada uno de estos cuadrantes se reconoce por sus coordenadas (fila y columna). Además, cada cuadrante se divide también como un tablero de ajedrez en 8 por 8 sectores.

La sala de control del Aventura contiene los siguientes elementos:

• Una pantalla de radar de corto alcance, que presenta un diagrama del cuadrante en que se encuentra la nave. Los objetos situados en dicho cuadrante vienen representados por los siguientes símbolos:

<=>Aventura	–O– Base estelar
+++ Nave enemiga	* Estrella

• Otro diagrama (pantalla de largo alcance) muestra las condiciones en los cuadrantes fronterizos con el que ocupa el Aventura, en forma de tabla de 3 filas y 3 columnas. El cuadrante donde nos encontramos aparece en el centro de esta tabla. La información de cada cuadrante está codificada de la siguiente forma: la cifra de las unidades es el número de estrellas en el cuadrante, la de las decenas es el número de bases estelares y la de las centenas el de naves enemigas. Por ejemplo, el número 217 significa que en el cuadrante correspondiente hay 7 estrellas, una base estelar y dos naves enemigas, mientras el número 4 significa que hay 4 estrellas, ninguna base estelar y ninguna nave enemiga.

• A la derecha de la pantalla aparece un informe condensado de la situación actual, que contiene las siguientes informaciones útiles:

1. Fecha estelar. Permite comprobar el paso del tiempo y llevar cuenta de cuánto tiempo nos queda para cumplir la misión.
2. Condición. Puede ser «normal (verde)», «alerta amarilla» (energía insuficiente), «alerta roja» (enemigos a la vista) o «en puerto» (el Aventura está amarrado a una base estelar).
3. Cuadrante. Coordenadas (fila, columna) del cuadrante de la galaxia donde se encuentra el Aventura.
4. Sector. Coordenadas (fila, columna) del sector de dicho cuadrante donde se encuentra el Aventura.
5. Energía total de que disponemos en cada momento.
6. Número de torpedos que tenemos a nuestra disposición.
7. La parte de la energía total actualmente desviada para actuar como escudo protector contra los ataques enemigos.



La pantalla del juego de la «Guerra espacial» presenta la sala de control de la nave Aventura. Hay una imagen de radar de corto alcance, un diagrama de largo alcance y una zona de información de las condiciones del vuelo.

Utilizando todos estos datos, el jugador debe decidirse por una de las siguientes acciones, que se ejecutan presionando teclas adecuadas:

1. Navegación. Es preciso elegir curso y velocidad. El curso es un número de 1 a 9, que define una dirección en la rosa de los vientos, de acuerdo con el diagrama siguiente:

```
    4 3 2
    \ | /
5 — + — 1
    / | \
    6 7 8
```

Se pueden dar cursos enteros o decimales (1.5 es una dirección intermedia entre 1 y 2); 9 es la misma dirección que 1.

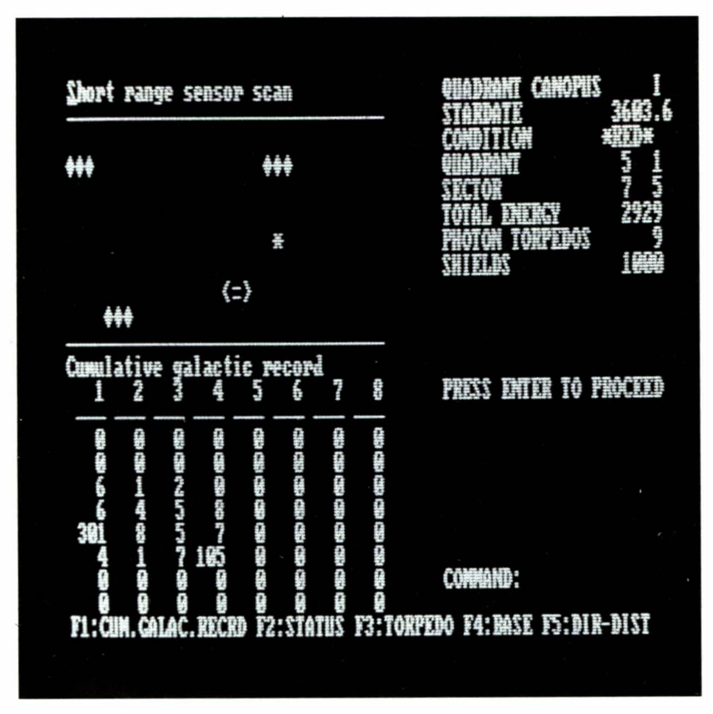
La velocidad es un número entre 0 y 8. Un valor de 1 hace pasar al Aventura a la misma posición o sector en un cuadrante contiguo (dependiendo de la dirección elegida).

2. Cañones láser. Permiten atacar simultáneamente a todos los enemigos presentes en el mismo cuadrante que el Aventura. Pero ellos también disponen de este tipo de cañones y pueden devolver el golpe.

3. Torpedos. Se lanzan en una dirección determinada, dada por un número de 1 a 9 (ver Navegación). Si el enemigo es alcanzado, es destruido instantáneamente y no puede responder, pero sí lo hará si el torpedo falla. Es posible utilizar el computador de a bordo para calcular el curso adecuado para los torpedos.

4. Escudo de protección. Parte de la energía de la nave puede dedicarse a formar un campo de fuerza que protege contra los cañones láser del enemigo. No hay que olvidar poner en marcha el escudo si hay peligro de combate. El dato ENERGIA TOTAL, dado en la pantalla, incluye también la energía del escudo.

5. Control de daños. Se obtiene el estado de reparación de todos los dispositivos. Un valor de cero indica que no hay daños. Un valor negativo, que el dispositivo está dañado, tanto más cuanto mayor sea el valor absoluto.



La nave Aventura dispone de una computadora que permite obtener información sobre la estructura general de la galaxia, calcular la trayectoria de los torpedos y otras operaciones complejas.

6. Computadora de a bordo. Permite obtener información adicional. Dispone de cinco opciones, que se seleccionan con las mismas teclas:

a) Mapa acumulado de la galaxia. Muestra la memoria de la computadora de todas las pantallas de largo alcance obtenidas hasta el momento, distribuidas sobre un tablero de ocho filas y ocho columnas, donde cada casilla representa uno de los cuadrantes de la galaxia.

b) Informe de situación. Indica cuántas fechas estelares nos quedan para cumplir la misión, cuántas naves enemigas existen aún y cuántas bases estelares hay en la galaxia.

c) Cálculo de dirección de los torpedos. Responde con la dirección y distancia a que se encuentra de nosotros cada una de las naves enemigas situadas en el cuadrante en que nos encontramos.

d) Cálculo de la dirección de la base estelar. Nos indica la dirección y la distancia a que se encuentra de nosotros la base estelar situada en el mismo cuadrante que el Aventura, si la hay.

e) Cálculo de la dirección y distancia a que se encuentra cualquier cuadrante de la galaxia. Esta opción es útil para ayudar en la navegación de la nave.

Por último, existe una opción que permite abandonar el juego definitivamente.

El listado completo del juego es el siguiente:

Programa 3.2: Juego de la guerra espacial en lenguaje BASIC.

Este programa es válido para IBM PC o compatibles. No sería muy difícil adaptarlo para AMSTRAD, pero al suponer que la pantalla tiene 80 columnas es de difícil adaptación para SPECTRUM o COMMODORE.

```
 10 REM Juego de la guerra espacial. Por M. Alfonseca.
 20 REM Basado en la versión de M. Mayfield
 30 DIM G(8,8): REM Mapa de la Galaxia
 40 DIM QQ$(8,32): REM Mapa de un cuadrante de la galaxia
 50 DIM ZZ(10,10): REM Mapa acumulado de la galaxia
 60 DIM D(8): REM Daños en la nave Adventure
 70 DIM K(10,3): REM Naves enemigas en este cuadrante
 80 DIM B(1,2): REM Bases estelares en este cuadrante
 90 DIM E(8,2): REM Estrellas en este cuadrante
 99 REM Variables C utilizadas por TORPEDO
 100 DIM CA(8),CB(8),CC(8),CD(8)
 110 CA(1)=0: CA(2)=-1: CA(3)=-1: CA(4)=-1: CA(5)=0: CA(6)=1: CA(7)=1:
CA(8)=1
 120 CB(1)=-1: CB(2)=0: CB(3)=0: CB(4)=1: CB(5)=1: CB(6)=0: CB(7)=0:
CB(8)=-1
 130 CC(1)=1: CC(2)=1: CC(3)=0: CC(4)=-1: CC(5)=-1: CC(6)=-1: CC(7)=0:
CC(8)=1
```

```
140 CD(1)=0: CD(2)=-1: CD(3)=-1: CD(4)=0: CD(5)=0: CD(6)=1: CD(7)=1:
CD(8)=0
500 KEY OFF: FOR I=1 TO 10: KEY I,"": NEXT I
1000 RANDOMIZE: REM Comienza la creación de una aventura espacial
1010 B9=0: K9=0: REM Generación de estrellas, bases y naves enemigas
1020 FOR I=1 TO 8: FOR J=1 TO 8: A=RND
1030 AK=0
1040 IF A>.8 THEN AK=1
1050 IF A>.95 THEN AK=2
1060 IF A>.9799999 THEN AK=3
1070 K9=K9+AK
1080 A=RND
1090 AB=0
1100 IF A>.96 THEN AB=1
1110 B9=B9+AB
1120 AS=1+INT(8*RND)
1130 G(I,J)=AS+10*AB+100*AK
1140 NEXT J: NEXT I
1150 K7=K9
1200 REM Posición de la nave AVENTURA
1210 Q1=1+INT(8*RND): Q2=1+INT(8*RND)
1220 S1=1+INT(8*RND): S2=1+INT(8*RND)
1230 REM Fechas
1240 T0=100*(20+INT(20*RND)): REM  Fecha inicial
1250 T=T0: REM  Fecha actual
1260 T9=30: IF K9>20 THEN T9=35: REM  Fecha final
1270 REM Otros datos
1280 E0=3000: REM Energía inicial
1290 E1=E0: REM Energía actual
1300 P0=10: REM Número inicial de torpedos
1310 P1=P0: REM Número actual de torpedos
1320 S0=0: REM Escudo protector actual
1500 REM Comienza la aventura
1510 PRINT "Al capitán de la nave AVENTURA. Estas son las órdenes:"
1520 PRINT "Debe usted destruir las";K9;"naves enemigas"
1530 PRINT "que han invadido la Galaxia, antes de que puedan"
1540 PRINT "atacar el cuartel general de la Federación"
1550 PRINT "en la fecha estelar";T0+T9;". Esto le deja";T9;"fechas."
1560 PRINT "Hay";B9"bases estelares en la Galaxia, donde puede"
1570 PRINT "reabastecer la nave AVENTURA. iBuena suerte!"
1580 PRINT: INPUT "Presione ENTER cuando esté dispuesto a asumir el
mando.";X
1600 CLS: REM Inicializar pantalla
1610 LOCATE 2,6
1620 PRINT "Pantalla de corto alcance"
1650 LOCATE 3,44
1660 PRINT "FECHA ESTELAR"
1670 LOCATE 4,44
1680 PRINT "CONDICION"
1690 LOCATE 5,44
```

```
1700 PRINT "CUADRANTE"
1710 LOCATE 6,44
1720 PRINT "SECTOR"
1730 LOCATE 7,44
1740 PRINT "ENERGIA TOTAL"
1750 LOCATE 8,44
1760 PRINT "TORPEDOS"
1770 LOCATE 9,44
1780 PRINT "ESCUDO PROTECTOR"
2000 REM Sala de control
2010 GOSUB 2020: GOTO 2200
2020 LOCATE 13,6: PRINT "Pantalla de largo alcance   ": LOCATE 14,6
2030 IF D(3)<0 THEN 2150
2040 PRINT "____________"
2050 FOR I=1 TO 3: LOCATE 14+I,6: FOR J=1 TO 3
2060 A=0
2065 A1=Q1+I: A2=Q2+J
2070 IF (A1>2) AND (A1<11) AND (A2>2) AND (A2<11) THEN A=G(A1-2,A2-2)
2080 ZZ(Q1+I-1,Q2+J-1)=A: PRINT USING "### ";A;
2090 NEXT J: NEXT I
2100 LOCATE 18,6: PRINT "____________": RETURN
2150 GOSUB 12600: PRINT "Pantalla de largo alcance estropeada"
2160 FOR I=1 TO 5: LOCATE 13+I,6: FOR J=1 TO 3
2170 PRINT "            "
2180 NEXT J: NEXT I: RETURN
2200 REM Entramos en un nuevo cuadrante
2210 LOCATE 5,60
2220 PRINT Q1;Q2
2230 S3=G(Q1,Q2): K3=INT(S3/100): S3=S3-100*K3: B3=INT(S3/10):
S3=S3-10*B3
2240 REM Distribución de objetos en el cuadrante
2250 FOR I=1 TO 8: FOR J=1 TO 32: QQ$(I,J)=" ": NEXT J: NEXT I
2260 QQ$(S1,1+4*(S2-1))="<"
2270 QQ$(S1,2+4*(S2-1))="="
2280 QQ$(S1,3+4*(S2-1))=">"
2290 F1$="+": F2$="+": F3$="+"
2300 FOR I=1 TO K3: GOSUB 2400: K(I,1)=R1: K(I,2)=R2: K(I,3)=200: NEXT I
2303 IF K3=3 THEN 2310
2305 FOR I=K3+1 TO 3: K(I,3)=0: NEXT I
2310 F1$="-": F2$="O": F3$="-"
2320 FOR I=1 TO B3: GOSUB 2400: B(I,1)=R1: B(I,2)=R2: NEXT I
2330 F1$=" ": F2$="*": F3$=" "
2340 FOR I=1 TO S3: GOSUB 2400: E(I,1)=R1: E(I,2)=R2: NEXT I
2350 GOTO 2500
2400 R1=1+INT(8*RND): R2=1+INT(8*RND)
2410 IF QQ$(R1,2+4*(R2-1))<>" " THEN 2400
2420 QQ$(R1,1+4*(R2-1))=F1$
2430 QQ$(R1,2+4*(R2-1))=F2$
2440 QQ$(R1,3+4*(R2-1))=F3$
2450 RETURN
```

```
2500 REM Comprobación de posición de amarre con estación
2510 IF B3=0 THEN 2600: REM No hay base
2520 IF S2<>B(1,2) THEN 2600: REM No estamos amarrados
2530 IF (S1<B(1,1)-1) OR (S1>B(1,1)+1) THEN 2600
2540 LOCATE 4,60: PRINT "EN PUERTO        "
2550 D0=1: REM En puerto
2560 E1=E0: P1=P0: S0=0: REM  Repostamos
2570 GOSUB 12600: PRINT "Sin escudo protector para el amarre"
2580 GOTO 2750
2600 D0=0: REM  No estamos amarrados
2610 IF K3>0 THEN 2650
2620 IF E1<.1*E0 THEN 2700
2630 LOCATE 4,60: PRINT "NORMAL (VERDE)"
2640 GOTO 2750
2650 LOCATE 4,60: COLOR 2: PRINT "ALERTA ROJA    ": COLOR 7
2660 PLAY "O3T150L8MSAAAP8AAA"
2670 IF S0>200 THEN 2750
2680 GOSUB 12600: PRINT "Escudo protector peligrosamente bajo"
2690 GOTO 2750
2700 LOCATE 4,60: COLOR 14: PRINT "ALERTA AMARILLA": COLOR 7
2750 IF D(2)<0 THEN 2800: REM Pantalla de corto alcance
2760 LOCATE 3,6: PRINT "________________________________"
2770 FOR I=1 TO 8: LOCATE 3+I,6
2780 FOR J=1 TO 32: PRINT QQ$(I,J);: NEXT J: NEXT I
2790 GOTO 2820
2800 GOSUB 12600: PRINT "Pantalla de corto alcance estropeada"
2810 FOR I=1 TO 8: LOCATE 3+I,6: FOR J=1 TO 32: PRINT " ";: NEXT J: NEXT
I
2820 LOCATE 12,6: PRINT "________________________________"
2830 LOCATE 3,60: PRINT T
2840 LOCATE 6,60: PRINT S1;S2
2850 LOCATE 7,60: PRINT E1+S0;"   "
2860 LOCATE 8,60: PRINT P1
2870 LOCATE 9,60: PRINT S0;"   "
2900 REM Comprobación de energía suficiente
2910 IF S0+E1<10 THEN 2930
2920 IF (E1>10) OR (D(7)=0) THEN 3000
2930 PLAY "O3T150L8MSAAAP8AAA"
2940 CLS: PRINT "¡¡¡ERROR FATAL!!! La nave ha quedado varada en el
espacio."
2950 PRINT "No hay energía suficiente, y el control de escudo no puede"
2960 PRINT "pasar energía a la sala de máquinas."
2970 GOTO 15000
2999 REM Bucle de petición de órdenes
3000 LOCATE 22,44: PRINT "ORDENES:                "
3010 LOCATE 23,44: PRINT "                        "
3020 LOCATE 24,6
3030 PRINT "F1: NAV F2: CAÑ F3: TOR F4: ESC F5: DAÑ F6: COM F10: FIN";
3040 GOSUB 12000: REM Lee una tecla
3060 IF I=68 THEN CLS: GOTO 15010: REM F10: Fin
```

```
3070 IF I>64 THEN 3040
3080 IF I=59 THEN 4000
3090 IF I=60 THEN 5000
3100 IF I=61 THEN 6000
3110 IF I=62 THEN 7000
3120 IF I=63 THEN 8000
3130 IF I=64 THEN 9000
4000 REM Control de navegación
4010 LOCATE 22,44: PRINT "CURSO (1-9):         ";
4020 N1=1: N2=9: LOCATE 22,65: GOSUB 12100: IF N1=-1 THEN 3000 ELSE
C1=N1
4030 LOCATE 23,44: PRINT "VELOCIDAD (0-8):  ";
4040 N1=0: N2=8: LOCATE 23,65: GOSUB 12100: IF N1=-1 THEN 3000 ELSE
W1=N1
4050 IF C1=9 THEN C1=1
4060 IF (W1=<.2) OR (D(1)>=0) THEN 4100
4070 GOSUB 12600: PRINT "MOTORES DAÑADOS                    "
4080 LOCATE 15,44: PRINT "VELOCIDAD MAXIMA = 0.2            "
4090 GOTO 4000
4100 N=INT(.5+8*W1): IF E1>=N THEN 4150
4110 GOSUB 12600: PRINT "ENERGIA INSUFICIENTE PARA MANIOBRAR "
4120 IF (D(7)<0) OR (S0<N-E1) THEN 3000
4130 LOCATE 15,44: PRINT S0;"UNIDADES DISPONIBLES EN EL ESCUDO"
4140 GOTO 3000
4150 REM Comienza el movimiento
4160 REM Primero el enemigo ataca
4170 GOSUB 12300
4180 REM Reparaciones en marcha
4190 GOSUB 12500
4200 XX1=S1: XX2=S2
4210 QQ$(INT(S1),1+4*(INT(S2)-1))=" "
4220 QQ$(INT(S1),2+4*(INT(S2)-1))=" "
4230 QQ$(INT(S1),3+4*(INT(S2)-1))=" "
4240 X1=CA(INT(C1))+CB(INT(C1))*(C1-INT(C1))
4250 X2=CC(INT(C1))+CD(INT(C1))*(C1-INT(C1))
4260 I=1
4270 S1=S1+X1: S2=S2+X2
4280 IF (S1<1) OR (S1>=9) OR (S2<1) OR (S2>=9) THEN 4400
4290 IF QQ$(INT(S1),2+4*(INT(S2)-1))=" " THEN 4340
4300 S1=S1-X1: S2=S2-X2
4310 GOSUB 12600: PRINT"PARADA AUTOMATICA DE MOTORES          "
4320 LOCATE 15,44: PRINT"DEBIDO A MALA NAVEGACION"
4330 GOTO 4350
4340 I=I+1: IF N>=I THEN 4270
4350 QQ$(INT(S1),1+4*(INT(S2)-1))="<"
4360 QQ$(INT(S1),2+4*(INT(S2)-1))="="
4370 QQ$(INT(S1),3+4*(INT(S2)-1))=">"
4380 GOSUB 4900
4390 S1=INT(.5+S1): S2=INT(.5+S2): GOTO 2500
4400 XX1=(8*Q1)+XX1+N*X1
```

```
4410 XX2=(8*Q2)+XX2+N*X2
4420 Q1=INT(XX1/8): S1=INT(XX1-8*Q1)
4430 Q2=INT(XX2/8): S2=INT(XX2-8*Q2)
4440 IF S1=0 THEN S1=8: Q1=Q1-1
4450 IF S2=0 THEN S2=8: Q2=Q2-1
4460 IF Q1<1 THEN Q1=1: GOSUB 4800
4470 IF Q2<1 THEN Q2=1: GOSUB 4800
4480 IF Q1>8 THEN Q1=8: GOSUB 4800
4490 IF Q2>8 THEN Q2=8: GOSUB 4800
4500 GOSUB 4900
4510 S1=INT(.5+S1): S2=INT(.5+S2): GOTO 2000
4800 GOSUB 12600: PRINT "DETENCION AUTOMATICA DE MOTORES"
4810 LOCATE 15,44: PRINT "PERMISO DENEGADO PARA ABANDONAR"
4820 LOCATE 16,44: PRINT "LA GALAXIA"
4830 RETURN
4900 E1=E1-N-10
4910 IF E1>=0 THEN 4960
4920 GOSUB 12600: PRINT "ENERGIA TRANSFERIDA DESDE EL ESCUDO"
4930 LOCATE 15,44: PRINT "PARA COMPLETAR LA MANIOBRA"
4940 S0=S0+E1: IF S0<0 THEN S0=0
4950 E1=0
4960 TT1=.1*INT(10*W1): IF TT1>1 THEN TT1=1: T=T+TT1
4970 IF T>T0+T9 THEN 15000 ELSE RETURN
5000 REM Disparo de cañones láser
5010 IF K3>0 THEN 5040
5020 GOSUB 12600: PRINT "NO HAY NAVES ENEMIGAS A LA VISTA     "
5030 GOTO 3000
5040 IF D(4)>=0 THEN 5070
5050 GOSUB 12600: PRINT "EL CAÑON LASER NO FUNCIONA           "
5060 GOTO 3000
5070 GOSUB 12600: PRINT "CAÑON LASER APUNTANDO AL OBJETIVO   "
5090 LOCATE 15,44: PRINT "ENERGIA DISPONIBLE =";E1
5100 IF D(8)>=0 THEN 5120
5110 LOCATE 16,44: PRINT "MENOR PRECISION POR FALLO COMPUTADOR"
5120 LOCATE 22,44: PRINT "ENERGIA DE DISPARO: ";
5130 N1=0: N2=E1: LOCATE 22,65: GOSUB 12100: IF N1=-1 THEN 3000 ELSE
C1=N1
5140 E1=E1-C1
5150 GOSUB 12300
5160 IF D(7)>=0 THEN 5180
5170 C1=C1*RND
5180 C1=INT(C1/K3)
5190 FOR I=1 TO 3
5200 IF K(I,3)=0 THEN 5260
5210 H=INT((2+RND)*C1/SQR(((K(I,1)-S1)*(K(I,1)-S1))+((K(I,2)-S2)*
(K(I,2)-S2))))
5220 IF H=<.15*K(I,3) THEN 5260
5230 K(I,3)=K(I,3)-H
5240 GOSUB 12600: PRINT H;"UNIDADES PARA ENEMIGO EN";K(I,1);K(I,2)
5250 IF K(I,3)<0 THEN XA=K(I,1): XB=K(I,2): GOSUB 12200
```

```
 5260 NEXT I: GOTO 2500
 6000 REM Lanzamiento de un torpedo
 6010 GOSUB 12600
 6020 IF D(5)<0 THEN PRINT "LOS TUBOS DE TORPEDOS NO FUNCIONAN ":
GOTO 3000
 6030 IF P1=<0 THEN PRINT "TORPEDOS AGOTADOS                ": GOTO
3000
 6040 LOCATE 22,44: PRINT "CURSO TORPEDO (1-9):";
 6050 N1=1: N2=9: LOCATE 22,65: GOSUB 12100: IF N1=-1 THEN 3000
 6060 IF N1=9 THEN N1=1
 6070 X1=CA(INT(N1))+CB(INT(N1))*(N1-INT(N1))
 6080 X2=CC(INT(N1))+CD(INT(N1))*(N1-INT(N1))
 6090 E1=E1-2: P1=P1-1: XA=S1: XB=S2
 6100 LOCATE 23,44: PRINT "CURSO DEL TORPEDO:"
 6110 XA=XA+X1: XB=XB+X2
 6120 IF (XA>=9) OR (XB>=9) OR (XA<1) OR (XB<1) THEN 6300
 6130 LOCATE 23,65: PRINT XA;XB
 6140 I$=QQ$(INT(XA+.5),2+4*INT(XB-.5))
 6150 IF I$=" " THEN 6310
 6160 IF I$="+" THEN GOSUB 12200: GOTO 6250
 6170 IF I$="*" THEN 6290
 6180 GOSUB 12600: PRINT "¡BASE ESTELAR DESTRUIDA, ESTUPIDO!"
 6190 B3=B3-1: B9=B9-1: D0=0
 6200 QQ$(INT(XA),1+4*INT(XB-1))=" "
 6210 QQ$(INT(XA),2+4*INT(XB-1))=" "
 6220 QQ$(INT(XA),3+4*INT(XB-1))=" "
 6230 G(Q1,Q2)=S3+10*(B3+10*K3)
 6240 GOSUB 2020
 6250 GOSUB 12300
 6260 LOCATE 22,65: PRINT"              "
 6270 LOCATE 23,65: PRINT"              "
 6280 GOTO 2500
 6290 GOSUB 12600: PRINT"UNA ESTRELLA ABSORBIO EL TORPEDO": GOTO
6250
 6300 GOSUB 12600: PRINT"EL TORPEDO HA FALLADO EL OBJETIVO": GOTO
6250
 6310 IF XH<>0 THEN LOCATE 3+XH,6+XK: PRINT " "
 6320 XH=INT(XA+.5): XK=2+4*INT(XB-.5)
 6330 LOCATE 3+XH,6+XK: PRINT "": PLAY "MFP4MB": GOTO 6110
 7000 REM Traspaso de energía al escudo protector
 7010 GOSUB 12600
 7020 IF D(7)>=0 THEN 7030
 7025 GOSUB 12600: PRINT "EL ESCUDO PROTECTOR NO FUNCIONA": GOTO
3000
 7030 LOCATE 22,44: PRINT "ENERGIA AL ESCUDO: ";
 7040 N1=0: N2=E1+S0: LOCATE 22,65: GOSUB 12100: IF N1=-1 THEN 3000
 7050 E1=E1+S0-N1
 7060 S0=N1
 7070 GOSUB 12600: PRINT "ESCUDO A";S0;"SEGUN SUS ORDENES"
 7080 LOCATE 9,60: PRINT S0
```

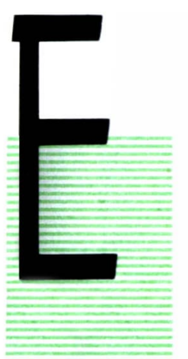
```
7090 LOCATE 22,65: PRINT "                ": GOTO 3000
8000 REM Informe de daños
8010 GOSUB 12600
8020 IF D(6)<0 THEN PRINT "EL CONTROL DE DAÑOS NO FUNCIONA    ":
GOTO 8110
8030 FOR I=1 TO 8
8040 LOCATE 13+I,44: GOSUB 8900: PRINT DD$;D(I)
8050 NEXT I: GOSUB 12700
8110 IF D0=0 THEN 8300
8120 D3=0: FOR I=1 TO 8: IF D(I)<0 THEN D3=D3+.1
8130 NEXT I: IF D3>1 THEN D3=1
8140 IF D3=0 THEN 8300
8150 GOSUB 12600: PRINT "TECNICOS DISPUESTOS A REPARAR NAVE"
8160 LOCATE 15,44: PRINT "TIEMPO DE REPARACION ESTIMADO:"
8170 LOCATE 16,44: PRINT D3;"FECHAS ESTELARES"
8180 LOCATE 17,44: PRINT "¿AUTORIZA USTED LAS REPARACIONES?"
8190 LOCATE 22,44: PRINT "SI/NO:        "
8200 LOCATE 22,65: INPUT I$: IF I$<>"SI" THEN 2500
8210 FOR I=1 TO 8: D(I)=0: NEXT I
8230 T=T+D3+.1: GOTO 8030
8300 GOSUB 2020: GOSUB 2020: FOR I=1 TO 8: LOCATE 13+I,44
8310 PRINT "                                  ": NEXT I
8320 GOTO 2500
8900 ON I GOTO 8910,8920,8930,8940,8950,8960,8970,8980
8910 DD$="MOTORES              ": RETURN
8920 DD$="PANTALLA CORTO AL.   ": RETURN
8930 DD$="PANTALLA LARGO AL.   ": RETURN
8940 DD$="CAÑONES LASER        ": RETURN
8950 DD$="TUBOS DE TORPEDOS    ": RETURN
8960 DD$="CONTROL DE DAÑOS     ": RETURN
8970 DD$="CONTROL DE ESCUDO    ": RETURN
8980 DD$="COMPUTADORA          ": RETURN
9000 REM Computadora de a bordo
9010 GOSUB 12600
9020 IF D(8)<0 THEN 9150
9030 PRINT "COMPUTADOR ACTIVO, ESPERANDO ORDENES"
9040 LOCATE 24,6
9050 PRINT "F1: GALAXIA F2: SITUACION F3: TORPEDOS F4: BASE F5:
DIR-DIST";
9060 GOSUB 12000 REM Lee una tecla
9070 IF I>63 THEN 9060
9080 IF I=59 THEN 9200
9090 IF I=60 THEN 9400
9100 IF I=61 THEN 9500
9110 IF I=62 THEN 9700
9120 IF I=63 THEN 9800
9150 PRINT "LA COMPUTADORA NO FUNCIONA          ": GOTO 3000
9200 REM Mapa de la galaxia
9210 LOCATE 13,6: PRINT "Mapa acumulado de la Galaxia"
9220 LOCATE 14,6: PRINT "  1  2  3  4  5  6  7  8"
```

```
9230 LOCATE 15,6: PRINT "________________________________"
9240 FOR I=1 TO 8: LOCATE 15+I,2: PRINT I: LOCATE 15+I,6
9250 FOR J=1 TO 8: PRINT USING " ###";ZZ(I+1,J+1);: NEXT J: NEXT I
9260 LOCATE 14,44: PRINT "Presione una tecla para seguir      "
9270 A$=INKEY$: IF A$="" THEN 9270
9280 LOCATE 14,6: PRINT "                                  "
9290 LOCATE 15,6: PRINT "                                  "
9300 FOR I=1 TO 8: LOCATE 15+I,2:
9310 PRINT "                                  ": NEXT I
9320 LOCATE 14,44: PRINT "                              "
9330 GOSUB 2020: GOTO 3000
9400 REM Situación
9410 GOSUB 12600: PRINT "Informe de Situación:"
9420 LOCATE 15,44: PRINT "Quedan";K9;"naves enemigas"
9430 LOCATE 16,44: PRINT "Quedan";T0+T9-T;"fechas estelares"
9440 LOCATE 17,44: PRINT "Hay";B9;"bases estelares en la Galaxia"
9450 GOTO 3000
9500 REM Cálculo de dirección para torpedos
9505 GOSUB 12600
9510 IF K3=0 THEN PRINT "No hay naves enemigas aquí": GOTO 3000
9512 FOR I=1 TO 3
9515 IF K(I,3)=0 THEN 9525
9520 N1=K(I,1): N2=K(I,2): GOSUB 9530
9525 NEXT I: GOTO 3000
9528 REM Rutina de cálculo de dirección/distancia
9530 LOCATE 14,44: PRINT " DIRECCION    DISTANCIA"
9540 X1=N1: X2=N2
9550 A=S1-X1: X=X2-S2
9560 IF X<0 THEN 9610 ELSE IF A<0 THEN 9640
9570 IF X>0 THEN 9580 ELSE IF A=0 THEN 9620
9580 C1=1
9590 IF ABS(A)>ABS(X) THEN C1=C1+2-ABS(X/A) ELSE C1=C1+ABS(A/X)
9600 GOTO 9670
9610 IF A>0 THEN 9630 ELSE IF X=0 THEN 9640
9620 C1=5: IF (A=0) AND (X=0) THEN 9670 ELSE 9590
9630 C1=3: GOTO 9650
9640 C1=7
9650 IF ABS(A)<ABS(X) THEN C1=C1+2-ABS(A/X) ELSE C1=C1+ABS(X/A)
9670 LOCATE 14+I,44: PRINT C1,SQR(X*X+A*A): RETURN
9700 REM Cálculo de dirección para bases
9710 GOSUB 12600
9720 IF B3=0 THEN 9790
9730 N1=B(1,1): N2=B(1,2): I=1: GOSUB 9530: GOTO 3000
9790 PRINT "No hay bases estelares aquí": GOTO 3000
9800 REM Cálculo de dirección cualquiera
9810 LOCATE 22,44: PRINT "1Ø COORDENADA:";
9820 LOCATE 23,44: PRINT "2Ø COORDENADA:";
9830 N1=1: N2=8: LOCATE 22,65: GOSUB 12100: IF N1=-1 THEN 3000
9835 X1=N1: N1=1: N2=8: LOCATE 23,65: GOSUB 12100: IF N1=-1 THEN
3000
```

```
9840 N2=N1: N1=X1: IF N1=9 THEN N1=1
9850 IF N2=9 THEN N2=1
9860 X2=N2: A=Q1-X1: X=X2-Q2
9870 GOSUB 12600: PRINT " DIRECCION    DISTANCIA"
9880 I=1: GOSUB 9560: GOTO 3000
12000 REM Lee una tecla de función
12010 A$=INKEY$: IF A$="" THEN 12010: IF LEN(A$)<>2 THEN 12010
12020 I=ASC(RIGHT$(A$,1)): IF I<59 THEN 12010
12030 RETURN
12100 REM Petición de datos
12110 INPUT A
12130 IF (A<N1) OR (A>N2) THEN 12160
12150 N1=A: N2=B: RETURN
12160 GOSUB 12600: PRINT"VALOR INVALIDO": N1=-1: RETURN
12200 REM Nave enemiga destruida
12210 PLAY "O3L8MNT120CDE2"
12220 GOSUB 12600: PRINT"¡NAVE ENEMIGA DESTRUIDA!"
12230 K3=K3-1: K9=K9-1
12240 IF K9=0 THEN 12280
12250 FOR I3=1 TO 3: IF XA<>K(I3,1) THEN 12270
12260 IF XB=K(I3,2) THEN 12271
12270 NEXT I3
12271 K(I3,3)=0
12272 QQ$(INT(XA+.5),1+4*INT(XB-.5))=" "
12273 QQ$(INT(XA+.5),2+4*INT(XB-.5))=" "
12274 QQ$(INT(XA+.5),3+4*INT(XB-.5))=" "
12275 G(Q1,Q2)=S3+10*(B3+10*K3)
12276 GOSUB 2020: RETURN
12280 CLS: PRINT "¡Enhorabuena, capitán! La última nave enemiga"
12285 PRINT "ha sido destruida": PRINT
12290 PRINT "Su eficiencia es igual a";INT(1000*K7/(T-T0)): END
12300 REM Ataque enemigo
12310 IF K3=0 THEN RETURN
12320 IF D0=0 THEN 12340
12330 GOSUB 12600: PRINT"LA BASE PROTEGE LA NAVE AVENTURA":
RETURN
12340 FOR I2=1 TO 3: IF K(I2,3)=0 THEN 12460
12350 H=SQR(((K(I2,1)-S1)*(K(I2,1)-S1))+((K(I2,2)-S2)*(K(I2,2)-S2)))
12360 H=INT((2+RND)*K(I2,3)/H)
12370 IF H=0 THEN 12460
12380 S0=S0-H: GOSUB 12600: PRINT H;"UNIDADES ALCANZAN AL
AVENTURA"
12390 LOCATE 15,44: PRINT "DESDE EL SECTOR ";K(I2,1);K(I2,2)
12400 IF S0<0 THEN CLS: PRINT"EL AVENTURA HA SIDO DESTRUIDO": GOTO
15000
12410 LOCATE 16,44: PRINT "ESCUDO DISMINUYE A ";S0
12420 IF (H<20) OR (.02>=H/S0) OR (RND>.5) THEN 12460
12430 I=INT(1+RND*8): D(I)=D(I)-(H/S0)-.5*RND
12440 LOCATE 17,44: PRINT "CONTROL DE DAÑOS INFORMA:"
12450 LOCATE 18,44: GOSUB 8900: PRINT DD$;" DAÑADO"
```

```
12460 GOSUB 12700: NEXT I2: RETURN
12500 REM Reparaciones en marcha
12505 D1=14: D6=W1: IF D6>1 THEN D6=1
12510 FOR I=1 TO 8: IF D(I)>=0 THEN 12530
12515 D(I)=D(I)+D6: IF D(I)<0 THEN 12530
12516 IF D(I)>0 THEN D(I)=0
12520 LOCATE D1,44: GOSUB 8900
12525 PRINT DD$;" REPARADO  ": GOSUB 12700
12530 NEXT I
12535 IF RND>.1 THEN RETURN
12540 I=1+INT(8*RND)
12545 D(I)=D(I)-.1-RND*5
12550 GOSUB 8900
12560 GOSUB 12600: PRINT"CONTROL DE DAÑOS INFORMA:"
12565 LOCATE 15,44: PRINT DD$;" DAÑADO    "
12590 RETURN
12600 REM Borra zona de información
12610 FOR J=1 TO 8: LOCATE 22-J,44
12620 PRINT "                              ": NEXT J
12630 LOCATE 14,44: RETURN
12700 REM Retardo
12710 FOR I=1 TO 500: J=2.2*I: NEXT I: RETURN
15000 PRINT "Es la fecha estelar ";T
15010 PRINT "Quedan";K9;"naves enemigas en la Galaxia al final de su
misión."
15020 PRINT "¡La federación será conquistada!"
```

JUEGOS DE BUSQUEDA DE TESOROS 4

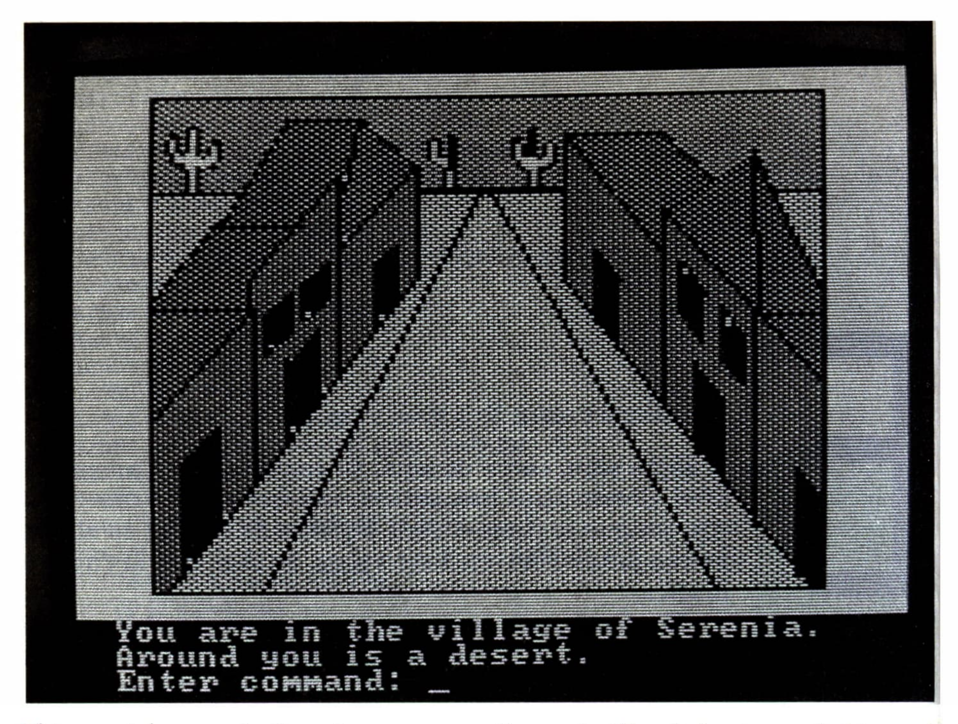


N algún sitio, cerca de aquí, se encuentra la Caverna Colosal, donde otros han encontrado fortunas en oro y tesoros, aunque se rumorea que algunos de los que en ella penetran jamás vuelven a ser vistos. Se dice que la cueva es mágica.

El párrafo anterior es la introducción de «Adventure», uno de los juegos de búsqueda de tesoros más antiguos, inventado por W. Crawther y modificado por Don Woods, de la Universidad de Stanford, en California. Aunque no fue precisamente el primero, su enorme popularidad a través de los años transcurridos desde su creación ha convertido a este juego en el paradigma o ejemplo de todos los juegos de ordenador que pertenecen a la misma clase.

En general, los juegos de búsqueda de tesoros tienen un contenido bastante simple y que se puede resumir en unas pocas palabras: el «aventurero» (llamaremos así a la persona que se sienta ante el teclado para jugar a este tipo de juegos de ordenador) debe explorar cierto territorio (una caverna formada por muchas cámaras o una casa con muchas habitaciones interconectadas) y buscar cierto número de tesoros, de los que debe apoderarse y que a veces ha de transportar hasta un lugar fijo. Durante su marcha debe tener cuidado de escapar de ciertas trampas o de vencer a algunos enemigos que tratarán de robarle, atacarle y (en sentido metafórico) matarle. Generalmente se obtiene al final del juego una puntuación o calificación que depende de los tesoros que el aventurero ha podido acumular, de los enemigos que ha destruido, o de ambas cosas a la vez. El verdadero objetivo del juego es obtener la calificación más alta posible.

Es obvio que el juego será tanto más atractivo cuanto más extenso sea el territorio que hay que explorar y cuanto mayor sea el número de tesoros, enemigos y trampas que puede uno encontrarse. La caverna en la ver-



El juego «Adventure in Serenia» es una versión semigráfica de los juegos de búsqueda de tesoros. El diálogo entre el jugador y el programa sigue siendo totalmente verbal.

sión más popular de «Adventure» [1] comprende algo menos de 80 cámaras, junto con dos laberintos que añaden otras 37 encrucijadas y callejones sin salida, lo que da un total de 116 lugares diferentes. Quince tesoros y alrededor de una docena de objetos útiles están esparcidos por la cueva. El aventurero puede encontrarse con cinco tipos de enemigos: una serpiente gigante, un dragón, un troll, un pirata y un número no determinado de enanos hostiles. En comparación con esto, el mundo subterráneo de «Zork» [2], un juego algo posterior de la misma clase, contiene 191 lugares, más de 200 objetos y muchos enemigos diferentes.

Todos los juegos de búsqueda de tesoros son, por supuesto, interactivos (no habría juego si el aventurero no pudiera hacer nada). Los más antiguos tienen la particularidad de que la comunicación entre el jugador y el ordenador se lleva a cabo por medio de un diálogo en lenguaje casi natural. El programa hace el papel de «narrador activo», pues describe al

[1] «Microsoft Adventure», IBM Personal Computer Entertainment Series, Implemented by G. Letwin, June 1981.

[2] Lebling, P.D.; Blank, M.S.; Anderson, T.A. «Zork: a computerized fantasy simulation game», IEEE Computer, 1979.

aventurero el territorio que le rodea en cada momento, le indica los objetos que están a su alcance en un instante dado y trata de llevar a cabo sus instrucciones, proporcionándole información sobre las consecuencias de éstas. Por otra parte, el jugador puede pedirle al programa información que desconoce o que ha olvidado o puede indicarle que realice ciertas acciones en nombre suyo (como moverse en una dirección determinada, recoger o abandonar ciertos objetos, o bien usar uno de éstos para algún propósito concreto.

No hace falta aclarar que los juegos de ordenador de este tipo no comprenden realmente el lenguaje natural. Todas las respuestas del programa son frases prefabricadas, cuya necesidad ha sido prevista por el programador del juego. Sin embargo, el análisis de la parte del diálogo que corresponde al aventurero presenta mayores dificultades, puesto que, como es obvio, es impredecible. Para permitir la mayor libertad posible al jugador, estos programas realizan un análisis de las frases que aquél escribe en el teclado, tratando de localizar palabras importantes, como verbos de acción o peticiones de información, nombres de objetos, personajes o lugares, direcciones de movimiento, etc., ignorando las palabras adicionales. Si el significado de la frase queda claro después de este análisis, el programa actúa en consecuencia inmediatamente. En caso contrario puede pedir información adicional, o puede dar una respuesta tal como «No lo entiendo». Como es lógico, cuanto mayor sea el diccionario de palabras que reconoce el programa, más se parecerá el diálogo a una conversación corriente.

Veamos un ejemplo correspondiente al juego «Collar», que puede clasificarse dentro de la misma clase que «Adventure» y «Zork»:



SESION DE EJEMPLO DEL JUEGO «COLLAR»

¡Bienvenido al MUNDO MÁGICO! ¿Deseas instrucciones?

SÍ

Vás a desempeñar el papel de un héroe en un juego de fantasía.
Tu misión consistirá en la destrucción del COLLAR MÁGICO,
un objeto del mundo de las tinieblas fabricado originalmente
por un personaje enigmático a quien sólo conocerás por el nombre
de «LA BRUJA».
Debes llevar el COLLAR al «VOLCÁN DEL FUEGO», para destruirlo.
Sólo así acabará para siempre el poder de LA BRUJA.
Durante tus viajes puedes encontrar ciertos personajes, algunos
de los cuales, amigos tuyos, tratarán de ayudarte, mientras

que otros, enemigos tuyos, servidores de LA BRUJA, tratarán de ponerte dificultades o incluso de destruirte.
¡BUENA SUERTE EN TU EMPRESA! ¡EL DESTINO DEL MUNDO DEPENDE DE TU ÉXITO!
Debes recordar llevar comida y agua en tus viajes, y reponerlos periódicamente en lugares donde sospeches su existencia.
Si se te olvida, puedes morir por falta de alimentos.
Si deseas información adicional, escribe AYUDA.
ESTÁS EN EL PUNTO DE PARTIDA, CERCA DE UN TERRITORIO DE COLINAS.
LA CARRETERA ESTE-OESTE CRUZA EL PUEBLO. UN CAMINO VA HACIA EL SUR.
AQUÍ ESTÁN LOS SIGUIENTES OBJETOS:
UN COLLAR MÁGICO
UN TROZO DE PEDERNAL
UNA MOCHILA DE COMIDA
UNA BOTELLA DE AGUA
UNA OLLA
TU CRIADO MAX ESTÁ AQUÍ
TU AMIGO OLVEG ESTÁ AQUÍ
TOMA TODO
TOMADO: UN COLLAR MÁGICO
TOMADO: UN TROZO DE PEDERNAL
TOMADO: UNA MOCHILA DE COMIDA
TOMADO: UNA BOTELLA DE AGUA
TOMADO: UNA OLLA
TOMADO: MAX
TOMADO: OLVEG
ESTÁS EN EL PUNTO DE PARTIDA
INVENTARIO
Ahora mismo llevas lo siguiente:
UN COLLAR MÁGICO
UN TROZO DE PEDERNAL
UNA MOCHILA DE COMIDA
UNA BOTELLA DE AGUA
UNA OLLA
Las siguientes personas vienen contigo:
MAX
OLVEG
ESTÁS EN EL PUNTO DE PARTIDA
ESTE
ESTÁS CAMINANDO POR UNA CARRETERA QUE AQUÍ VA DE ESTE A OESTE.
EL CAMINO SE PIERDE EN LA DISTANCIA EN AMBAS DIRECCIONES.
Hay 1 monstruo verde en las proximidades

MATA AL MONSTRUO
No eres bastante fuerte para matar un MONSTRUO
ESTÁS EN LA CARRETERA ESTE-OESTE
¡CUIDADO CON LOS MONSTRUOS VERDES!
El monstruo verde se acerca

....

ESTÁS EN LA ORILLA OCCIDENTAL DE UN RÍO ANCHO Y CLARO.
HAY AQUÍ UN MUELLE DONDE PUEDE DESEMBARCAR UN FERRY.
EL FERRY ESTÁ EN ESTE LADO DEL RÍO. PARA USARLO, ESCRIBE «CRUZAR».
CRUZAR
SUBES AL FERRY Y CRUZAS EL RÍO.
ESTÁS EN LA ORILLA ESTE DEL RÍO CLARO.
HAY AQUÍ UN MUELLE DONDE PUEDE DESEMBARCAR UN FERRY.
EL FERRY ESTÁ EN ESTE LADO DEL RÍO. PARA USARLO, ESCRIBE «CRUZAR».

....

EL CAMINO QUE ESTÁS SIGUIENDO ACABA AQUÍ, CERCA DE UN ALTO SETO.
LAS COPAS DE NUMEROSOS ÁRBOLES APARECEN POR ENCIMA DEL SETO.
HAY UNA PUERTA PEQUEÑA BAJO EL SETO, QUE PARECE CRUZAR HASTA EL OTRO LADO.
CRUZAR
LA PUERTA ESTÁ CERRADA. NO PUEDES PASAR.
ESTÁS EN LA PUERTA BAJO EL SETO.
SUELTA LLAVES
SOLTADO: UN MANOJO DE LLAVES
ESTÁS EN LA PUERTA BAJO EL SETO.
AQUÍ ESTÁ EL SIGUIENTE OBJETO:
UN MANOJO DE LLAVES
ABRE LA PUERTA
¡No tienes las llaves!
ESTÁS EN LA PUERTA BAJO EL SETO.
AQUÍ ESTÁ EL SIGUIENTE OBJETO:
UN MANOJO DE LLAVES
TOMA LAS LLAVES
TOMADO: UN MANOJO DE LLAVES
ESTÁS EN LA PUERTA BAJO EL SETO.
ABRE LA PUERTA
O.K.
ESTÁS EN LA PUERTA BAJO EL SETO.
CRUZAR
CRUZAS BAJO EL SETO Y DESCUBRES

....

ESTÁS EN UN LUGAR TERRIBLE. HAY COLINAS ALREDEDOR CON TUMBAS EN TODAS PARTES. ES UN CEMENTERIO DE ANTIGUOS GUERREROS.
AQUÍ ESTÁ EL SIGUIENTE OBJETO:
UNA ESPADA
AQUÍ HAY UN TERRIBLE FANTASMA BLANCO
        MATA AL FANTASMA
No eres bastante fuerte para matar un FANTASMA
ESTÁS EN EL PÁRAMO MUERTO.
AQUÍ ESTÁ EL SIGUIENTE OBJETO:
UNA ESPADA
AQUÍ HAY UN TERRIBLE FANTASMA BLANCO
        DALE LA ESPADA AL FANTASMA
¡DEJA DE DECIR TONTERÍAS!
        TOMA LA ESPADA
El FANTASMA BLANCO no te deja tomar UNA ESPADA

....



ESTRUCTURA DE LOS JUEGOS DE BUSQUEDA DE TESOROS

Un juego de búsqueda de tesoros consta de tres componentes fundamentales:

1. Topografía.
2. Objetos.
3. Enemigos.



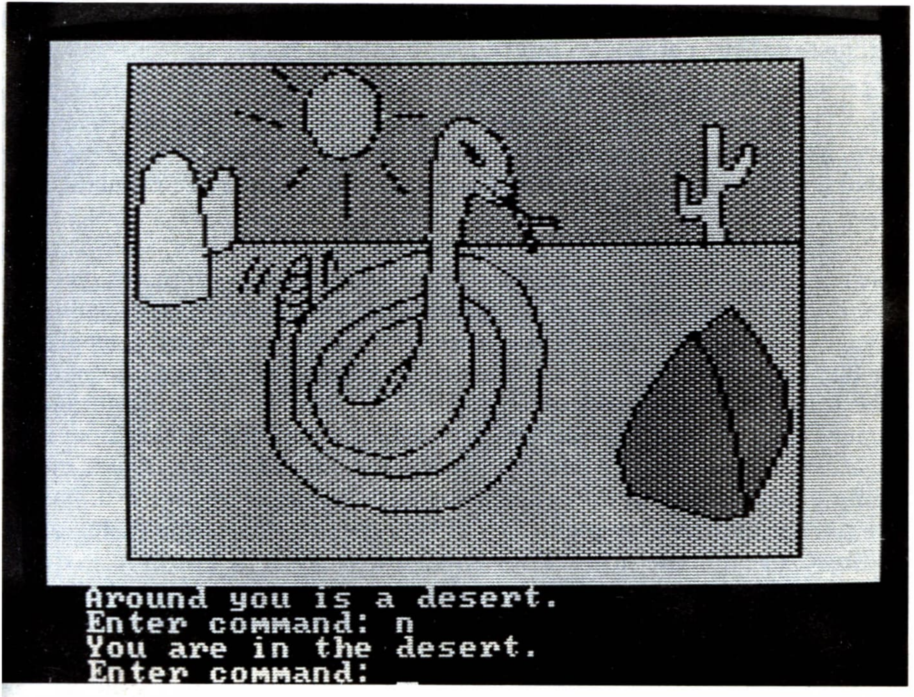
Topografía

Aunque los ambientes con los que nos encontramos en la vida real son continuos, la percepción humana introduce cierta división en partes más o menos separables. Desde el punto de vista del observador, las cosas que puede percibir directamente están comprendidas en una región limitada, más o menos centrada alrededor de él, y cuyo tamaño está generalmente asociado con el sentido de la vista. En el caso de un juego de ordenador es preciso dividir el territorio a explorar en zonas claramente definidas, eliminando el cambio continuo del ambiente que rodea en cada momento al aventurero. Por tanto, el territorio donde tiene lugar la aventura estará for-

mado por un conjunto de compartimientos bien definidos y separados entre sí. Estos compartimientos se denominan comúnmente «habitaciones» o «cámaras», y en cada juego existe un número determinado de ellos, más o menos grande según su complicación.

La región completa que comprende estas habitaciones es cerrada, es decir, el aventurero no puede salir de ella. Normalmente se la dota de límites formados por obstáculos concretos e infranqueables, tales como el mar (si el juego no permite al aventurero utilizar barcos), montañas inescalables o desiertos donde el aventurero se pierde y queda confuso respecto a la dirección a seguir, y de donde puede salir únicamente por la misma dirección por la que entró. Otras veces, como en el caso de una cueva, los obstáculos son más sólidos: las paredes de la caverna.

Los ríos son un caso especial. En general, no se pueden cruzar, excepto en uno o varios puntos particulares, donde existe un vado, un puente o un bote. En este último caso, además, se exige que el bote esté situado en el mismo lado del río que el aventurero y sus compañeros para que puedan cruzarlo. Por tanto, el juego debe mantener constancia de la posición en que se encuentran todos los botes en cada momento. Está claro que esto hace necesario que cada lado del río pertenezca a un compartimiento diferente.



Otra imagen de «Adventure in Serenia».

Cada «habitación» estará conectada con un máximo de diez habitaciones vecinas, correspondiendo cada conexión a una dirección diferente: norte, sur, este, oeste, nordeste, sudeste, noroeste, sudoeste, arriba y abajo. Sin embargo, una habitación determinada no tiene por qué estar unida a otras por todos los caminos posibles, puesto que obstáculos de varias descripciones (montañas, cursos de agua, terrenos difíciles o, simplemente, paredes) pueden hacer imposible moverse en una o varias direcciones. La mayor parte de las conexiones serán deterministas, es decir, cuando quiera que el aventurero trata de seguir el mismo camino a partir del mismo punto de partida, llegará al mismo destino. Sin embargo, unas pocas conexiones en la topografía pueden ser aleatorias, variando el punto de destino de partida en partida, o incluso en instantes diferentes dentro de la misma partida. El efecto probabilístico puede representar la confusión del aventurero en un momento dado, su pérdida del sentido de la dirección, movimientos reales de la topografía (por ejemplo, una habitación giratoria) o simples efectos mágicos.



Objetos

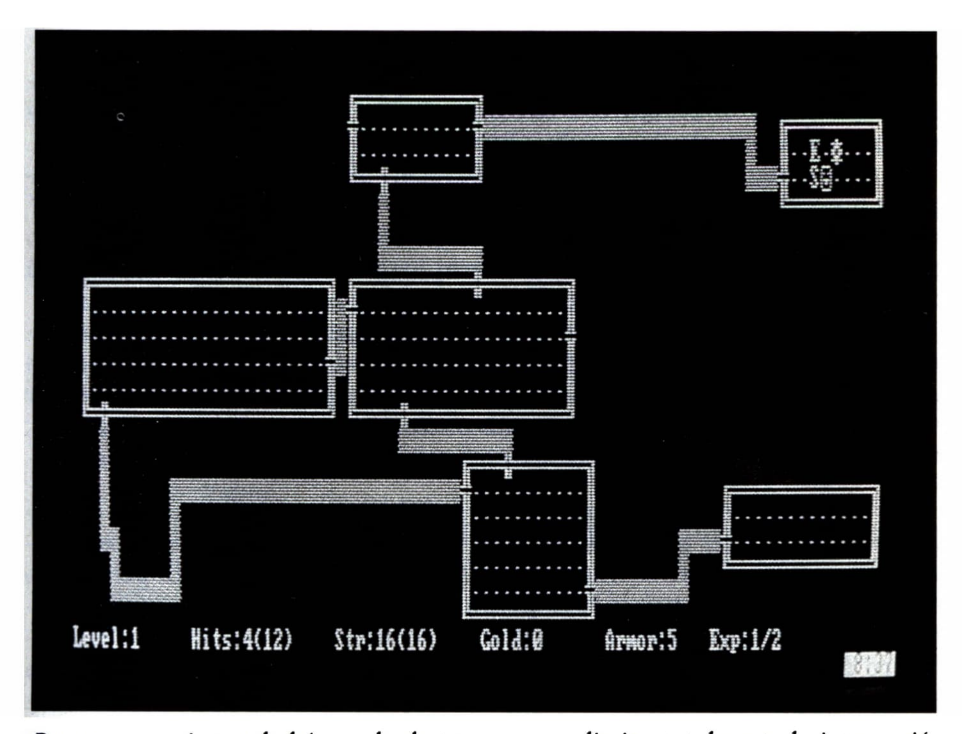
Los objetos que aparecen en un juego de búsqueda de tesoros pueden pertenecer a tres categorías diferentes, a saber:

1. Objetos inmóviles, como tumbas, losas de piedra con mensajes grabados, puertas, etc. Estos objetos aparecen ante los ojos del aventurero cuando éste pasa cerca de ellos, pero no puede llevárselos consigo. Sin embargo, a veces le proporcionan información útil para el desarrollo del juego.
2. Objetos móviles que el aventurero puede llevarse, como armas ofensivas y defensivas, comida, botellas de agua, tesoros y muchos otros objetos diversos. Algunos de ellos deben estar en posesión del aventurero para que éste pueda realizar acciones concretas. Por ejemplo, es preciso que disponga de un arma para que pueda atacar con éxito a un enemigo. Debe tener agua para poder beber, comida para comer, encantamientos mágicos para teleportarse de un lugar a otro, animales de carga para que le ayuden a llevar mayor número de objetos, etc. Otros no tienen misión alguna y se introducen en el juego para confundir al aventurero, haciéndole perder tiempo al recogerlos y disminuyendo sus posibilidades de llevar objetos realmente útiles. Pues en algunos juegos de aventura existe un cupo fijo de objetos que el jugador puede llevar consigo, y cuando lo alcanza no puede ya tomar otros nuevos sin soltar uno de los que llevaba previamente. En otros juegos, más sofisticados, cada objeto tiene un peso diferente, y el aventurero puede llevar una carga determinada, que corresponderá a un número de objetos más grande o más pequeño, dependien-

do del peso de cada uno. Además, tanto el peso de los objetos como la carga máxima que puede llevar el aventurero puede variar durante el desarrollo del juego. Por ejemplo, si el jugador está débil por falta de alimento o herido después de una batalla, no podrá llevar tanto peso como si está fuerte y en óptimas condiciones.

La comida y el agua constituyen un caso especial. Para poder llevarlos, el aventurero debe tener en su poder un recipiente adecuado (una mochila para la comida, una botella o pellejo para el agua). A medida que el jugador se mueve, la cantidad de comida y de agua de que dispone irá disminuyendo progresivamente, de manera que, si no lo repone, acabará por quedarse sin nada. Si el aventurero permite que se le acabe la provisión de alimento y de agua, irá quedándose cada vez más débil y, por tanto, no podrá llevar una carga tan grande como al principio, o se moverá más lentamente. Finalmente, puede llegar a morir (es decir, el juego termina) por falta de alimento.

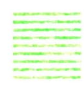
3. Una tercera categoría comprende los objetos que el jugador puede llevar consigo y, además, ponerse o quitarse sin abandonarlos. Por ejem-



«Rogue» es un juego de búsqueda de tesoros que elimina totalmente la interacción verbal y la sustituye por movimientos de teclas. También introduce elementos aleatorios que permiten que sea casi imposible que el jugador se encuentre dos veces en condiciones idénticas.

plo, una cota de malla que le proteja de los ataques enemigos o un anillo mágico que tenga ciertas propiedades deseables o indeseables (como volver invisible a quien lo lleva). Estos objetos pueden encontrarse, por tanto, en tres estados diferentes: sueltos, en poder del aventurero, pero sin que éste los lleve puestos explícitamente, o en su poder mientras está haciendo uso de ellos (el anillo puede estar en su bolsillo o en el dedo, la cota de malla puede estar en la mochila o sobre su camisa). El funcionamiento de cada objeto depende, como es natural, del estado en que se encuentra.

Todavía podríamos añadir a las tres clases anteriores una cuarta categoría, formada por los personajes amistosos que, si el jugador lo desea, pueden acompañarle y prestarle ayuda en el desempeño de su aventura.



Enemigos

En cuanto a los enemigos, pueden clasificarse también en varios grupos:

1. Peligros inmóviles, que se encuentran permanentemente en el mismo lugar y que el aventurero encontrará siempre que pase por él, a menos que sea capaz de destruirlos. Estos son los más corrientes.
2. Enemigos que pueden o no aparecer, al azar, o en momentos y lugares predefinidos. Los enanos de «Adventure» pertenecen a este grupo.
3. Personajes (como el ladrón de «ZORK») que tienen movimiento propio, independiente del que realice el jugador, que sólo los encontrará cuando coincida con ellos en el mismo lugar.
4. Personajes perseguidores, cuyo movimiento depende de los desplazamientos del jugador, y que tratan de acercarse a éste para atacarlo. Esto significa que este tipo de personajes acabará, más pronto o más tarde, por alcanzarlo. Si son varios, pueden reunirse a medida que el juego avanza, haciendo que el aventurero encuentre más difícil vencerlos.

En cualquier caso, siempre que el jugador se encuentra con un enemigo, debe responder al desafío de una forma adecuada. A veces puede volverse y escapar por el mismo camino por el que llegó, pero en general no se le permitirá atravesar la habitación o cámara sin luchar primero. Algunos enemigos pueden ser destruidos simplemente atacándolos, si el aventurero dispone de armas ofensivas o defensivas suficientes. Otros exigen la realización de acciones concretas y determinadas o la posesión de objetos especiales, y el jugador deberá descubrir la forma de eliminar a cada personaje antes de poder proseguir por el camino deseado. Por supuesto, todos estos encuentros tienen un elemento de riesgo, de manera que el aventurero puede resultar herido, o incluso muerto.

DIVERSOS JUEGOS DE BUSQUEDA DE TESOROS

Además de «Adventure» y «Zork», que ya hemos mencionado, que son bastante sencillos, han aparecido muchos otros juegos de esta familia que se basan más o menos directamente en «Adventure», pero añadiendo diversas complicaciones: gráficos simples «Combates y Tesoros», «Rogue»), gráficos complejos «King's Quest», «Adventure in Serenia»), movimientos de los personajes, introducción del azar en la disposición topográfica del territorio a explorar y la posición de los objetos y enemigos que contiene «Combates y Tesoros», «Rogue»). Esta última característica permite que sea posible jugar varias veces sin encontrarse siempre en las mismas situaciones, cosa que no ocurre en los juegos más antiguos y simples.

Como ejemplo de este tipo de juegos más complejos, voy a describir «Combates y Tesoros», desarrollado por mí y escrito en BASIC compilado, pues BASIC interpretado es demasiado lento para esta aplicación concreta.



COMBATES Y TESOROS

«Combates y Tesoros» es un generador automático de juegos de aventura. Cada uno de estos juegos proporciona un territorio que el jugador debe explorar, pues contiene numerosos tesoros ocultos que no es difícil descubrir y llevarse consigo. Pero también existen dificultades: cierto número de enemigos peligrosos, bandidos, ogros, e incluso dragones, se oponen a su marcha y están dispuestos a presentarle cara, e incluso a acabar con su exploración definitivamente. El objeto de cada juego consiste en explorar el territorio completo, apoderarse de todos los tesoros y vencer a todos los enemigos.

El desarrollo de un juego determinado puede interrumpirse en cualquier momento para continuarlo más tarde. También es posible crear varios juegos diferentes, aun cuando existan otros sin terminar. Cada juego tiene su propio territorio, distinto del de otros juegos. Los tesoros, enemigos y objetos de todo tipo estarán situados en lugares diversos. Esto proporciona a «Combates y Tesoros» una gran variedad, pues si un juego se termina con éxito puede comenzarse otro inmediatamente. Todas estas diferencias ayudan a mantener el interés de quien lo utiliza.

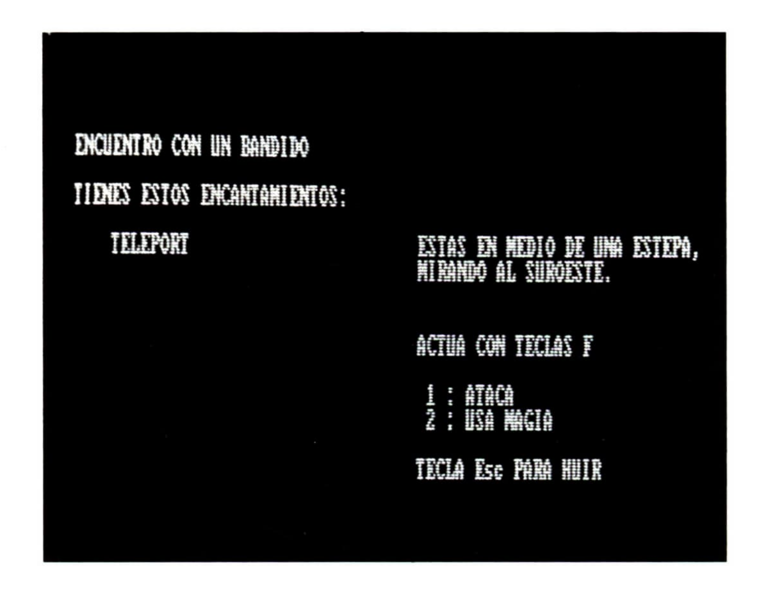
El número de juegos distintos que puede generar «Combates y Tesoros» es prácticamente ilimitado.



Cómo se juega

Cuando se llega a un lugar determinado, dentro del territorio de uno de los juegos generados por «Combates y Tesoros», pueden ocurrir tres cosas:

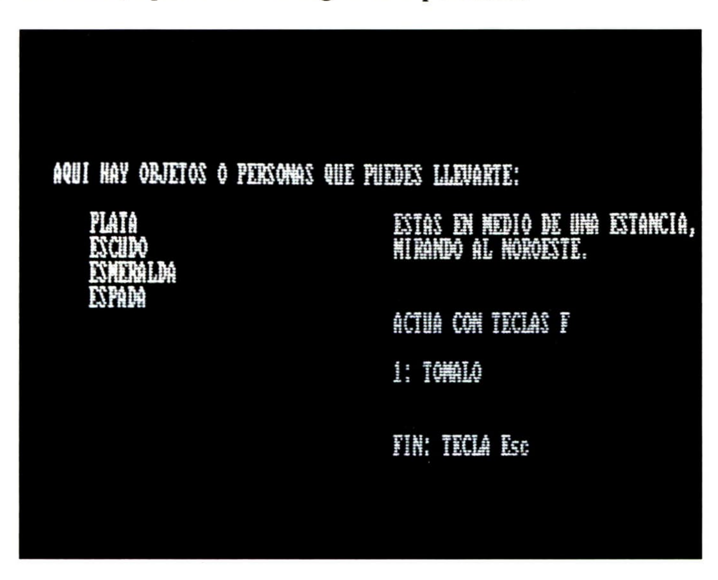
Primero. Hay un enemigo en ese mismo sitio. En este caso aparece una pantalla como la siguiente:



donde figuran el nombre del enemigo, el del lugar en donde se encuentra el jugador y la dirección hacia donde está mirando (norte, sur, este, oeste, nordeste, sudeste, noroeste o sudoeste). Ahora el jugador puede decidir (presionando la tecla correspondiente) entre atacar al enemigo, escapar volviendo por donde vino o transportarse a otro lugar, quizá muy lejano, utilizando una TELÉPORTación. Para poder optar a esto último debe tener en su poder algún encantamiento, que habrá encontrado previamente.

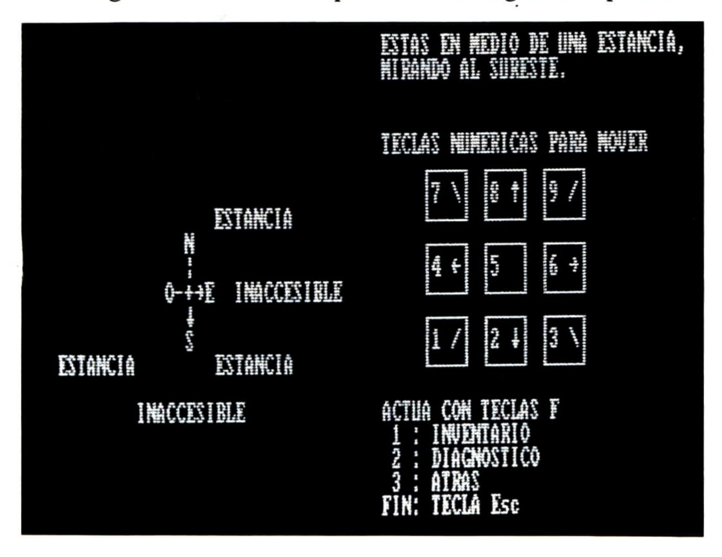
Si decide atacar, verá en la pantalla el resultado del ataque y la contestación del enemigo, si éste ha logrado sobrevivir. En este último caso, el jugador puede renovar su ataque o dar el combate por terminado utilizando una de las otras opciones.

Segundo. No hay un enemigo, pero sí un objeto o un personaje amistoso. En este caso, aparecerá la siguiente pantalla:



donde figuran los nombres de los objetos o personajes que se encuentran en ese lugar. Si el jugador desea llevarse consigo alguno de ellos, presionará la tecla indicada. Deberá entonces escribir una X a la izquierda de cada uno de los objetos que desea tomar.

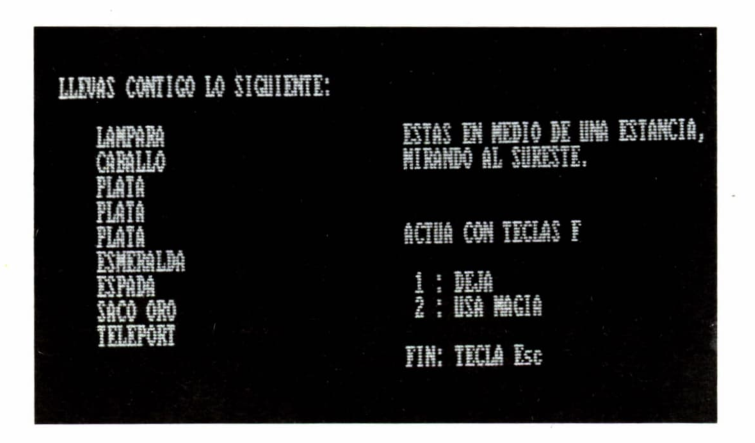
Tercero. No hay enemigos ni objetos o, habiendo objetos, no se desea llevárselos consigo. En este caso aparecerá la siguiente pantalla:



Esta pantalla contiene una brújula que indica la dirección en que está mirando el jugador y lo que puede distinguir de frente y a los lados, pero no a su espalda. Puede utilizar las teclas indicadas a la derecha para trasladarse a otro lugar cercano, escogiendo la dirección. Así, la tecla 8 le lleva hacia el norte, la tecla 2 hacia el sur, la tecla 1 hacia el sudoeste, etc.

También pueden utilizarse otras teclas para regresar por donde vino, para ver un inventario de los objetos que lleva consigo o para conocer la situación del juego: el estado de sus heridas (si acaba de realizar un combate) y los puntos que lleva acumulados.

Al pedir el inventario aparece la siguiente pantalla:



Ahora el jugador puede decidirse por abandonar algunos de los objetos que lleva consigo o de las personas que le acompañan. Aparecerá entonces el mensaje «TECLEE X ANTE POSICION DESEADA», permitiéndosele utilizar la tecla X para seleccionarlos, del mismo modo en que se hizo para tomarlos. También puede utilizarse una TELEPORTación para trasladarse rápidamente a otro lugar. De nuevo se le exigirá que el encantamiento se encuentre en su poder, antes de poder utilizarlo.



Cómo dejar de jugar

Para dar por terminada una sesión, temporal o definitivamente, debe presionarse la tecla correspondiente a FIN cuando se encuentre ante una pantalla de movimiento (la tercera del apartado anterior). Aparecerá entonces un resumen de la situación del juego (diagnóstico de las heridas y puntos que lleva acumulados), seguido del mensaje siguiente:

¿De veras quieres terminar? S/N

Si se presiona la tecla S aparecerá la siguiente pregunta:

¿Quieres guardar esta aventura para seguir luego? S/N

Si se desea abandonar esta sesión definitivamente, se presiona N. Si se va a continuar más tarde, se presiona S. En este último caso aparecerá la pregunta siguiente:

Nombre de la aventura:

Se escribe un nombre adecuado y se presiona la tecla ENTER. Esta aventura será guardada en el disquete o cinta y podrá reanudarse cuando se desee.



Estructura de «Combates y Tesoros»

Hemos visto que los elementos de un juego típico de búsqueda de tesoros pueden clasificarse en tres grupos principales: una topografía donde se mueve el aventurero, los objetos que encuentra y los enemigos que le atacan. En la práctica, las dos últimas clases pueden unirse en una sola, pues los enemigos no son otra cosa que objetos con propiedades especiales.

«Combates y Tesoros» ha sido desarrollado de acuerdo con las consideraciones anteriores. Es capaz de generar topografías relativamente aleatorias, y de distribuir objetos y personajes en las diferentes «habitaciones», permitiendo así al jugador cambiar no sólo sus actividades en la partida anterior, sino también la estructura misma de la partida.

Las unidades topográficas generadas automáticamente por el programa pertenecen a cuatro niveles de dificultad y a una de las categorías siguientes:

- Lugares puntuales, como cavernas, ciudades, castillos, cabañas y poblados. En un nivel determinado sólo puede existir un lugar único de cada uno de estos tipos.
- Superficies, como llanuras, bosques, junglas, campos de hielo, valles, cadenas montañosas. Estas unidades abarcan generalmente varias localidades topográficas consecutivas.
- Líneas, como ríos, caminos y carreteras. Estas ocupan varias posiciones consecutivas, componiendo configuraciones que se disponen de acuerdo con una dirección dominante. Estas unidades topográficas parten de lugares especiales (las carreteras de las ciudades o castillos, los caminos de los pueblos o cabañas, los ríos de las montañas) y terminan en unidades

de cierto tipo (las carreteras en ciudades o en otras carreteras, los ríos en el mar, otro río o un lago) o cuando alcanzan los límites del área del juego.

A cada tipo topográfico que pertenece a las dos primeras categorías se le asignan dos valores de probabilidad. El primero es la probabilidad básica de encontrar una unidad de ese tipo en una posición determinada. El segundo da la probabilidad de encontrar una unidad del mismo tipo en las unidades contiguas a la que se está generando. (Las unidades contiguas son aquéllas que están situadas a una distancia de una posición en cualquier dirección.)

Al principio del juego el aventurero se encuentra en el primer nivel, que corresponde a la exploración de un territorio al aire libre. Los tipos de unidades topográficas que aquí existen son los que se han indicado hasta ahora: ríos, lagos, llanuras, estepas, bosques, etc. Sin embargo, si el jugador penetra en una ciudad, pasa automáticamente al segundo nivel, como si la ciudad, que en el primer nivel ocupa un punto único, se hubiera expandido a su alrededor al penetrar en ella. Las unidades topográficas de la ciudad son calles y avenidas. Existe, además, una entrada. Al introducirse en ella, el jugador abandona la ciudad y vuelve al primer nivel. Otro tanto ocurre con el tercer nivel (el castillo, cuyas unidades topográficas son las habitaciones) y el cuarto (la caverna, compuesta de cierto número de cámaras interconectadas).

Los objetos pueden pertenecer a una de las categorías siguientes:

1. Comida
2. Agua
3. Tesoros
4. Armas ofensivas
5. Armas defensivas
6. Encantamientos (teleportaciones)
7. Objetos varios
8. Personajes amistosos
9. Enemigos fijos
10. Enemigos móviles

Cada tipo de objeto posee dos atributos que dan información sobre el objeto, dependiendo de la categoría a la que pertenece. Todos ellos tienen un peso. Además, los tesoros tienen un valor; las armas pueden ser más o menos poderosas; los enemigos más o menos duros, etc.

Cuando el jugador desea crear un nuevo entorno de juego, el programa produce automáticamente una topografía. El área del juego se considera como un conjunto de cuatro rectángulos numéricos (uno por nivel de juego) de lados de longitud aleatoria, dentro de ciertos límites prácticos. La fila y la columna de cada punto del rectángulo corresponden a las coordenadas geográficas simuladas, mientras que el valor del punto indi-

ca el tipo de «habitación». Existen más de veinte tipos de «habitaciones» diferentes, algunos de los cuales están reservados para niveles determinados.

Los lugares lineales (ríos, etc.) se generan posteriormente. Se selecciona un punto de partida adecuado, se escoge al azar una orientación general para la estructura y se utiliza ésta para calcular el índice de la posición contigua más próxima cuyo tipo va a reemplazarse por el de la estructura lineal que se está generando. Cuando se alcanza una posición de terminación adecuada, se detiene el proceso.

Finalmente, se distribuyen objetos y enemigos al azar por toda la topografía. Los enemigos más peligrosos y los tesoros más valiosos quedarán colocados en el cuarto nivel (la caverna), mientras que los enemigos más débiles y los tesoros menos valiosos pasarán a ocupar lugares en el primer nivel. No todos los tipos de lugares pueden contener objetos o enemigos. En particular, ninguno de ellos quedará en una posición inaccesible, aunque sí pueden asignárseles posiciones que sólo pueden alcanzarse mediante teleportaciones.

El programa que ejecuta la aventura procede mediante un bucle que comienza por analizar el contenido de la posición actual del jugador. Si hay un enemigo en esa posición, tiene lugar un encuentro. El jugador ataca primero, pero el enemigo puede responderle. El resultado de cada movimiento de ataque del aventurero depende de los siguientes factores:

- Las armas ofensivas del jugador.
- El grado de cansancio y heridas que tiene en cada momento.
- La fuerza del enemigo.
- Los puntos de experiencia que el jugador tiene acumulados como resultado de combates anteriores.
- Los compañeros que lleva consigo.

Los efectos de la respuesta del enemigo dependen de los siguientes factores:

- Las armas defensivas del jugador.
- El grado de cansancio y heridas que tiene en cada momento.
- La fuerza del enemigo.

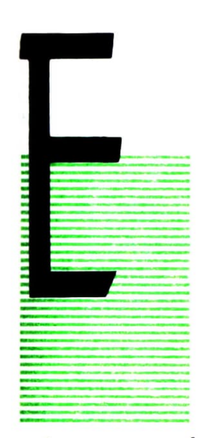
El resultado de un encuentro puede ser:

- Que el jugador sea muerto.
- Que el jugador decida huir por donde vino.
- Que el enemigo sea muerto.
- Que el enemigo huya (en cuyo caso cambia de posición y el jugador deberá volver a enfrentarse con él en otro momento y lugar).

Si el jugador ha vencido a los enemigos que ocupan una posición, o si no había ninguno, el programa comprueba si existe algún objeto en dicho lugar y ofrece al aventurero la posibilidad de tomarlo. Por último, se presenta la «perspectiva» de la topografía para dar al jugador la posibilidad de cambiar de lugar, de moverse de un sitio a otro. El programa guarda información de la orientación del jugador, además de su posición presente, y tiene esto en cuenta para generar la vista en «perspectiva».

Por último, el programa permite al jugador definir objetos y enemigos nuevos y dotarlos de sus propiedades correspondientes (valor si son tesoros, fuerza si son enemigos, peso si el aventurero puede llevárselos). Esta última característica permite aumentar aún más la versatilidad de las partidas que pueden efectuarse con «Combates y Tesoros».

CONSTRUYA SUS PROPIOS LIBRO-JUEGOS 5



N su cuento corto «Examen de la obra de Herbert Quain», incluido en el libro *El jardín de senderos que se bifurcan,* el escritor argentino Jorge Luis Borges describe un libro imaginario titulado *April March* (que corresponde, en inglés, a *Abril Marzo,* y no a la traducción igualmente posible *Marcha de Abril*). El libro está compuesto de trece capítulos. El primero describe los sucesos de cierto día. El segundo refiere los de la víspera (pues, como indica su título, *April March* es retrógado). Pero al llegar al tercero se interrumpe la sucesión temporal, pues lo que aquí se relata es otra versión de lo que ocurrió la víspera del primer capítulo. En cuanto al cuarto, presenta una tercera versión de los sucesos de la víspera.

Al llegar a los capítulos quinto, sexto y séptimo se produce un nuevo salto atrás en el tiempo, pues ahora se nos relatan tres versiones diferentes de lo que pasó antes del capítulo segundo. De igual manera, los capítulos octavo, noveno y décimo cuentan lo que sucedió el día anterior al capítulo tercero, e igual ocurre con los tres últimos capítulos con respecto al cuarto. El conjunto del libro contiene, pues, nueve novelas diferentes, todas de tres capítulos, dispuestas de acuerdo con el siguiente esquema:

5-2-1
6-2-1
7-2-1
8-3-1
9-3-1
10-3-1
11-4-1
12-4-1
13-4-1

Se observará que todas las novelas tienen en común el último capítulo, y que muchas de ellas comparten también el segundo. En cambio, el principio de cada una de las novelas es totalmente diferente del de las demás.

Que yo sepa, la novela retrógrada-ramificada de Herbert Quain no ha llegado a escribirse. Sin embargo, el concepto de ramificación ha sido incorporado en una forma nueva de literatura, aparecida recientemente por influencia de los juegos de ordenador de búsqueda de tesoros, y a la que el escritor argentino se adelantó en varias décadas: los «libro-juegos».

Un libro-juego es un libro que no se lee secuencialmente y que contiene en su interior muchas historias independientes. Cada una de sus divisiones, que normalmente abarca como máximo una o dos páginas (por lo que quizá es demasiado corta para llamarla «capítulo») termina ofreciendo al lector la posibilidad de tomar una decisión respecto al orden de su lectura, convirtiéndole así, aparentemente, en protagonista de una aventura propia y diferente. En realidad, un libro-juego contiene un número limitado de aventuras posibles, y es en esto equivalente a los juegos de búsqueda de tesoros de carácter fijo, que no introducen elementos aleatorios.

Veamos un ejemplo: en el libro-juego «Nuevo Viaje al Centro de la Tierra» (Ediciones Ingelek, 1986) al llegar al final de la página 9 encontramos las siguientes alternativas:

1. Si eliges el camino de la derecha, pasa a la página 19.
2. Si crees que Arne Saknussemm descendió por el del centro, pasa a la página 28.
3. En caso de que pienses que el camino adecuado es el de la izquierda, pasa a la página 77.

Si, por ejemplo, el lector se decide por la segunda opción, encontrará una nueva bifurcación después de leer dos páginas. Ahora se le dice:

1. Si eliges la galería del este, pasa a la página 15.
2. Si eliges la galería del oeste, pasa a la página 96.

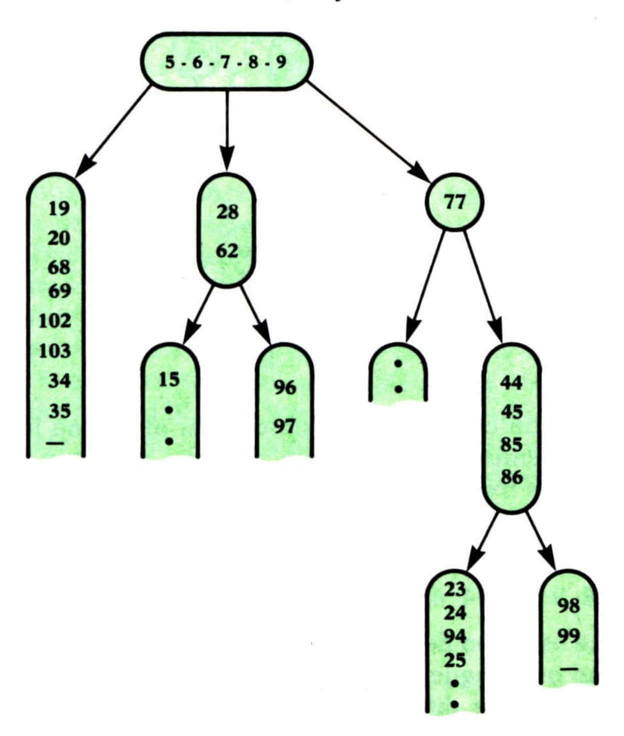
El conjunto del libro consta de dieciocho aventuras diferentes, a cada una de las cuales le corresponden las páginas siguientes:

1. 5 6 7 8 9 19 20 68 69 102 103 34 35
2. 5 6 7 8 9 28 62 15 100 87 95
3. 5 6 7 8 9 28 62 15 32 18 39 52 74 75 36 37 73
4. 5 6 7 8 9 28 62 15 32 18 39 52 74 75 36 37 76
5. 5 6 7 8 9 28 62 15 32 18 39 81 82
6. 5 6 7 8 9 28 62 15 32 18 39 22 90 91
7. 5 6 7 8 9 28 62 15 32 18 39 22 50 51 57 63 56
8. 5 6 7 8 9 28 62 15 32 18 39 22 50 51 58 59
9. 5 6 7 8 9 28 62 15 32 18 39 22 50 51 60 11 16 17
10. 5 6 7 8 9 28 62 15 32 18 39 22 50 51 60 11 64 12 14 34 35
11. 5 6 7 8 9 28 62 15 32 18 39 22 50 51 60 11 64 88 89

12. 5 6 7 8 9 28 62 96 97
13. 5 6 7 8 9 77 53 40 41 67
14. 5 6 7 8 9 77 53 40 41 48 49 54 55 83 84 42 43 70 72 46 47
15. 5 6 7 8 9 77 53 40 41 48 49 54 55 83 84 42 43 56
16. 5 6 7 8 9 77 44 45 85 86 23 24 94 25 30 66
17. 5 6 7 8 9 77 44 45 85 86 23 24 94 25 63 56
18. 5 6 7 8 9 77 44 45 85 86 98 99

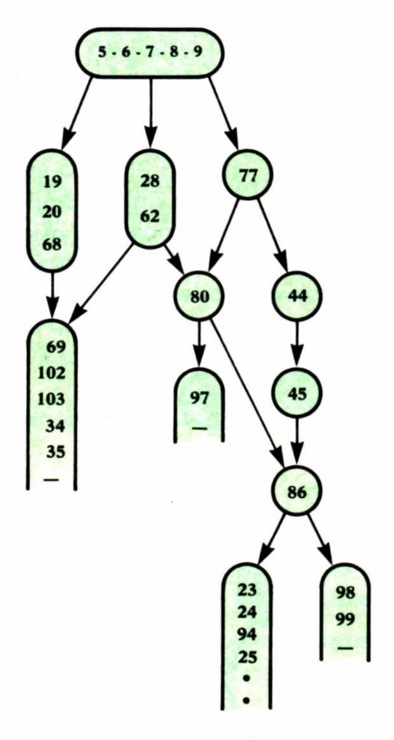
Como puede observarse, los dieciocho argumentos diferentes tienen en común las cinco primeras páginas, pero a partir de ese momento la acción se bifurca más o menos. Cuanto más lejos se produzca la bifurcación, más páginas idénticas tendrán las diferentes aventuras (véase el caso de los argumentos 14 y 15, que tienen comunes las primeras 17 páginas). Por otra parte, existen aventuras muy cortas (como la número 12, que tiene un total de nueve páginas) y otras más largas (como los números 10 y 14, que alcanzan las veintiuno).

Se observará, sin embargo, que en el libro-juego anterior dos aventuras ya no vuelven a reunirse una vez que se han separado. Esto significa que el conjunto de todos los caminos posibles que se pueden seguir al leer el libro puede representarse mediante una estructura arborescente. Veamos una parte de la que corresponde al ejemplo anterior, donde aparecen completos los caminos números 1, 12 y 18:



En conjunto, los libro-juegos de este tipo son novelas ramificadas parecidas a la inventada por Borges, pero no retrógradas. Es decir, la ramificación se produce al avanzar el tiempo, lo que es comprensible, pues de esta manera los libros son mucho más fáciles de construir.

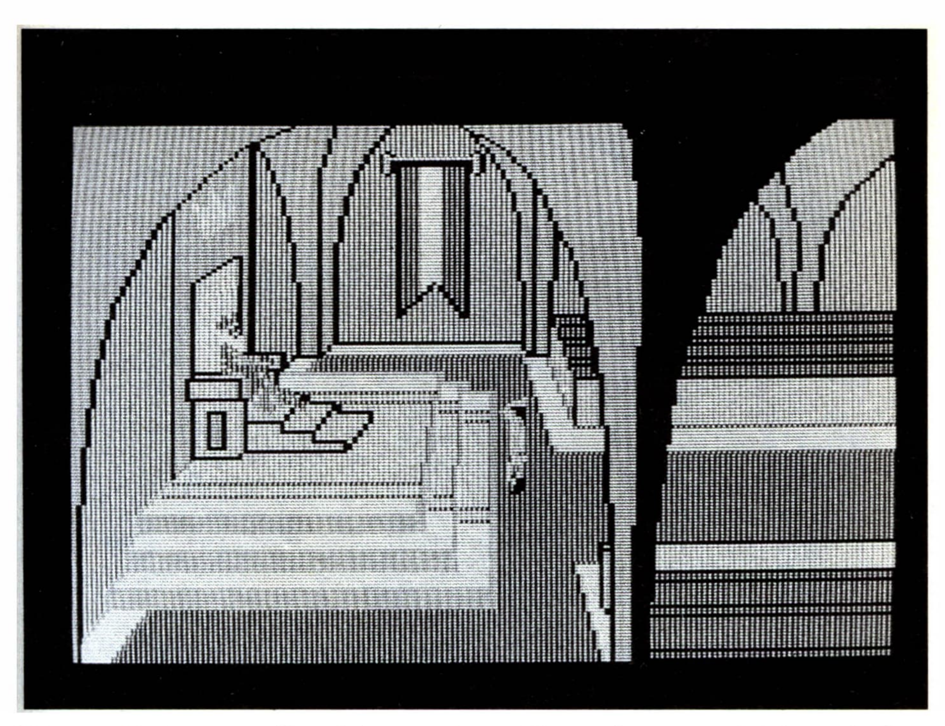
Otros libro-juegos contienen aventuras entrelazadas más complicadas, que pueden volver a reunirse una vez que se han separado, y que se parecen aún más a los juegos de búsqueda de tesoros. En estos libro-juegos la estructura formada por todos los caminos posibles que se pueden seguir no es un árbol, como en el caso anterior, sino un gráfico con circuitos como el siguiente:



En un gráfico con circuitos es posible llegar al mismo lugar por dos o más caminos diferentes. En el ejemplo anterior, el lector podría llegar a la página 86 por los tres caminos siguientes:

1. 5 6 7 8 9 28 62 80 86
2. 5 6 7 8 9 77 80 86
3. 5 6 7 8 9 77 44 45 86

Es obvio que este tipo de estructura presenta mayor dificultad de construcción, puesto que lo que ocurre en la página 86 tiene que ser compatible con las tres maneras diferentes de llegar a ella.

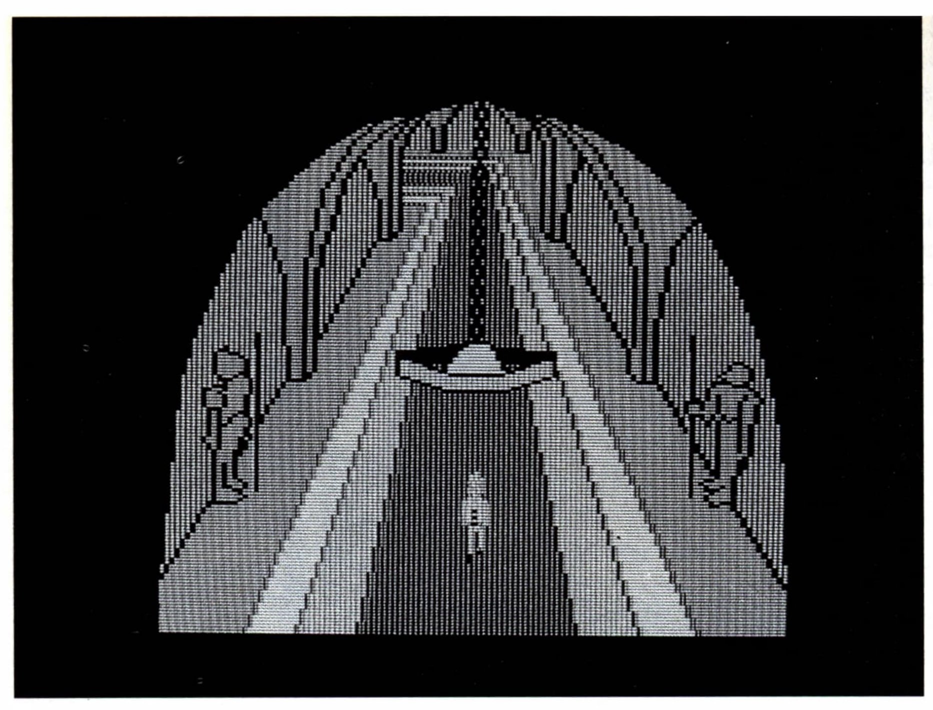


El juego «King's Quest» lleva la interacción gráfica en los juegos de búsqueda de tesoros hasta sus últimas consecuencias. El juego es animado y el jugador controla el movimiento de los dibujos mediante teclas.

Si los caminos fueran bidireccionales (por ejemplo, si la línea que conecta la página 86 con la página 80 pudiera recorrerse en los dos sentidos), el lector podría meterse en un bucle cerrado (es decir, leer las mismas páginas una vez tras otra). Esta es la situación normal y aceptable en un juego de búsqueda de tesoros, donde en general es posible y a veces útil recorrer varias veces el mismo camino dentro de la topografía del juego o desandar lo andado, pero no es posible en un libro-juego, porque la lectura de las páginas del libro es intrínsecamente unidireccional (cada página puede leerse sólo de arriba a abajo, pero no al revés). Esto significa que las líneas del gráfico de un libro-juego estarán siempre dirigidas en un sentido determinado, por lo que los bucles cerrados sólo serán posibles si el constructor del juego provoca a propósito la aparición de una condición como la siguiente:

1. 5 6 7 8 9 28 62 80 86 28

Es decir, que de la página 28 pueda saltarse a la 62, de ésta a la 80, luego a la 86 y, por último, otra vez a la 28. En general, nunca debe incluirse



Los dibujos de «King's Quest» son extraordinariamente detallados. El objetivo del juego es descubrir y conseguir tres tesoros determinados.

más de un bucle en un libro-juego. Encontrar uno puede proporcionarle al lector una agradable sorpresa, pero si se abusa de esto la lectura del libro resulta monótona.

Como puede observarse, el gráfico que presenta a la vista todos los caminos posibles de lectura en un libro-juego resulta muy semejante a la topografía de los juegos de ordenador de la clase descrita en el capítulo anterior, que en general puede representarse también mediante un gráfico Tanto es así, que resulta muy fácil construir un libro-juego a partir de la descripción de un juego de búsqueda de tesoros. Tan fácil, que es posible programar un ordenador para que pase de uno al otro automáticamente.



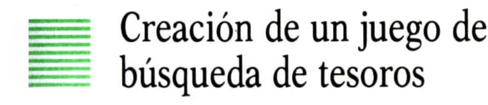
CONSTRUCCION AUTOMATICA DE LIBRO-JUEGOS

En las próximas páginas voy a presentar tres programas escritos en el lenguaje BASIC, que permiten realizar las siguientes aplicaciones:

1. Construir un número indefinido de juegos de búsqueda de tesoros bastante generales y tan complicados como se desee (programa HAZJUEGO).

2. Ejecutar cualquiera de los juegos creados por el programa anterior, es decir, jugar a ellos (programa JUEGO).

3. Convertir cualquiera de los juegos creados por HAZJUEGO en un libro-juego equivalente, que podrá observarse en la pantalla del ordenador o del que se podrá obtener una copia escrita en una impresora (programa VERJUEGO).



Creación de un juego de búsqueda de tesoros

Veamos, en primer lugar, el programa HAZJUEGO:

Programa 5.1. Programa HAZJUEGO en lenguaje BASIC.

Este programa es válido para SPECTRUM, AMSTRAD, COMMODORE e IBM PC o compatibles (ver notas).

```
1    REM Creación de un librojuego. Por M. Alfonseca.
10   DIM S$(300): DIM E$(800): DIM O$(100)
15   DIM CL(300): DIM IN(300)
20   DIM E1(300): DIM O1(300):
25   DIM V1(800): DIM V2(250): DIM V3(100): DIM V4(170)
30   DIM HB(100): DIM FZ(100): DIM PR(100): DIM CN(100)
40   LET I=0 : REM Indice de sitios
50   LET IE=0: REM Indice de caminos
60   LET IL=0: REM Indice de lista de caminos
70   LET IM=0: REM Indice de enemigos
80   LET IO=0: REM Indice de objetos
90   LET IJ=0: REM Indice de lista de objetos
95   LET IC=0: REM Indice de Condiciones
100  REM Bucle de creación
110  PRINT "Sitio numero ";I+1
120  PRINT "¿Qué te dicen?"
130  INPUT A$
140  IF A$="" THEN GOTO 9000
150  LET S$(I+1)=A$
160  PRINT "¿De qué clase es?"
170  PRINT "1: Elegir camino"
180  PRINT "2: Luchar"
190  PRINT "3: Obtener objetos"
200  PRINT "4: Comprobar objetos"
210  PRINT "5: Fin del juego"
220  INPUT CL(I+1)
230  ON CL(I+1) GOTO 1000,2000,3000,4000,5000
1000 REM Elegir camino
1010 LET IN(I+1)=IL+1
```

```
1020 LET K=0
1030 PRINT "Caso ";K+1
1040 INPUT B$
1050 IF B$="" THEN GOTO 1120
1060 LET E$(IE+1)=B$
1070 INPUT "¿A dónde va? ";A
1080 LET V1(IE+1)=A
1090 LET IE=IE+1
1100 LET K=K+1
1110 GOTO 1030
1120 LET E1(IL+1)=K
1130 LET IL=IL+1
1140 GOTO 6000
2000 REM Luchar
2010 LET IN(I+1)=IM+1
2020 INPUT "¿Habilidad del enemigo? ";HB(IM+1)
2030 INPUT "¿Fuerza del enemigo? ";FZ(IM+1)
2040 INPUT "¿A dónde vas si ganas? ";V2(3*IM+1)
2050 INPUT "¿A dónde vas si abandonas? ";V2(3*IM+2)
2060 INPUT "¿A dónde vas si pierdes? ";V2(3*IM+3)
2070 LET IM=IM+1
2080 GOTO 6000
3000 REM Obtener objetos
3010 LET IN(I+1)=IJ+1
3020 LET K=0
3030 PRINT "Objeto ";K+1
3040 INPUT B$
3050 IF B$="" THEN GOTO 3110
3060 LET O$(IO+1)=B$
3070 INPUT "¿Precio? ";PR(IO+1)
3080 LET IO=IO+1
3090 LET K=K+1
3100 GOTO 3030
3110 LET O1(IJ+1)=K
3120 INPUT "¿A dónde vas después? ";V3(IJ+1)
3130 LET IJ=IJ+1
3140 GOTO 6000
4000 REM Comprobar objetos
4010 LET IN(I+1)=IC+1
4020 PRINT "¿Qué objeto tienes que tener?"
4030 INPUT B$
4040 FOR K=1 TO IO
4050 IF O$(K)=B$ THEN GOTO 4090
4060 NEXT K
4070 PRINT "Error. Ese objeto no ha sido definido"
4080 GOTO 4020
4090 LET CN(IC+1)=K
4100 INPUT "¿A dónde vas si lo tienes? ";V4(2*IC+1)
4110 INPUT "¿A dónde vas si no lo tienes? ";V4(2*IC+2)
4120 LET IC=IC+1
```

```
4130 GOTO 6000
5000 REM Sitio final de juego
5010 LET IN(I+1)=0
6000 LET I=I+1
6010 GOTO 100
9000 REM Guardar el juego en cinta o disco
9010 INPUT "¿Nombre de la aventura? ";F$
9020 IF F$="" THEN GOTO 9999
9040 OPEN F$ FOR OUTPUT AS #1
9050 PRINT #1,I;IE;IL;IM;IO;IJ;IC
9060 FOR K=1 TO I
9070 PRINT #1,S$(K)
9080 NEXT K
9090 FOR K=1 TO I
9100 PRINT #1,CL(K);IN(K)
9110 NEXT K
9150 FOR K=1 TO IE
9160 PRINT #1,E$(K)
9170 NEXT K
9180 FOR K=1 TO IE
9190 PRINT #1,V1(K)
9200 NEXT K
9210 FOR K=1 TO IL
9220 PRINT #1,E1(K)
9230 NEXT K
9240 FOR K=1 TO IM
9250 PRINT #1,HB(K);FZ(K);V2(3*K-2);V2(3*K-1);V2(3*K)
9260 NEXT K
9270 FOR K=1 TO IO
9280 PRINT #1,O$(K)
9290 NEXT K
9300 FOR K=1 TO IO
9310 PRINT #1,PR(K)
9320 NEXT K
9330 FOR K=1 TO IJ
9340 PRINT #1,O1(K);V3(K)
9350 NEXT K
9360 FOR K=1 TO IC
9370 PRINT #1,CN(K);V4(2*K-1);V4(2*K)
9380 NEXT K
9390 CLOSE #1
9999 END
```

NOTAS: El programa anterior es válido directamente para IBM PC.

Para SPECTRUM cambiar las siguientes líneas:

```
10   DIM S$(300,15): DIM E$(800,8): DIM O$(100,8)
230  GOTO 1000*CL(I+1)
9999 STOP
```

Sustituir también las líneas comprendidas entre 9040 y 9390 por las siguientes:

```
9040 DIM A(7)
9050 LET A(1)=I
9060 LET A(2)=IE
9070 LET A(3)=IL
9080 LET A(4)=IM
9090 LET A(5)=IO
9100 LET A(6)=IJ
9110 LET A(6)=IC
9120 SAVE F$ DATA A()
9130 SAVE F$ DATA S$()
9140 SAVE F$ DATA E$()
9150 SAVE F$ DATA O$()
9160 SAVE F$ DATA CL()
9170 SAVE F$ DATA IN()
9180 SAVE F$ DATA E1()
9190 SAVE F$ DATA O1()
9200 SAVE F$ DATA V1()
9210 SAVE F$ DATA V2()
9220 SAVE F$ DATA V3()
9230 SAVE F$ DATA V4()
9240 SAVE F$ DATA HB()
9250 SAVE F$ DATA FZ()
9260 SAVE F$ DATA PR()
9270 SAVE F$ DATA CN()
```

Para AMSTRAD cambiar las siguientes líneas:

```
9040 OPENOUT F$
9390 CLOSEOUT
```

y cambiar todas las referencias a #1 por #9.

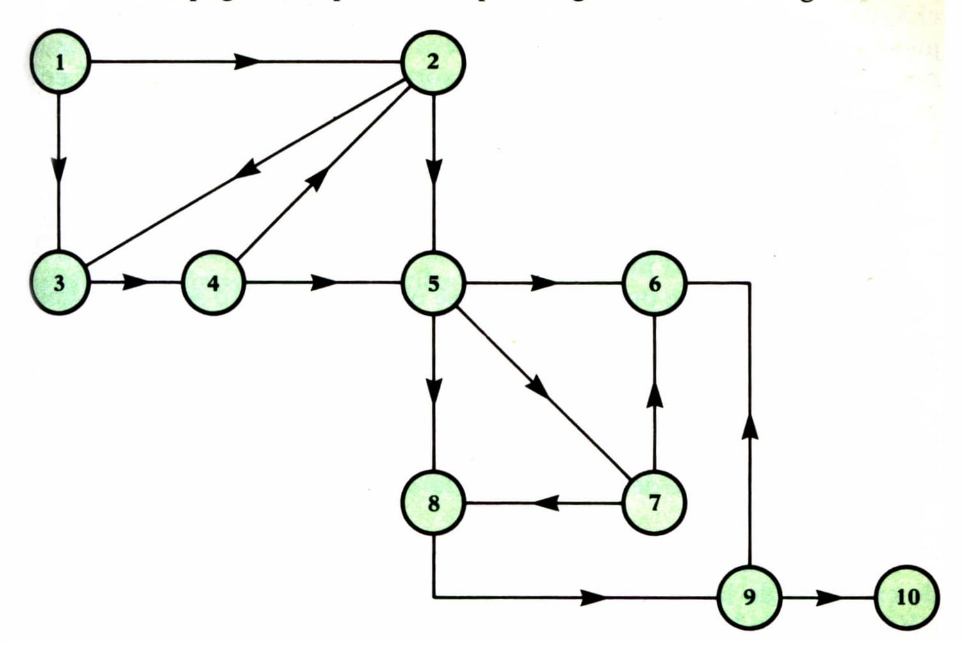
Para COMMODORE cambiar las siguientes líneas:

```
9040 OPEN 1,1,1,F$
9390 CLOSE 1
```

Programa 5.1. *Programa HAZJUEGO*

Antes de ejecutar el programa HAZJUEGO, será preciso que se defina el juego concreto que se desea construir. Dicho juego vendrá definido por los siguientes elementos (daré en cada caso un ejemplo ilustrativo).

1. Una topografía, representada por un gráfico como el siguiente:



2. El texto que encontrará el jugador al llegar a cada punto del gráfico (a cada «habitación», en la terminología de los juegos de búsqueda de tesoros). El programa HAZJUEGO permite dar textos de una línea como máximo.

3. Los objetos que puede encontrar el jugador, dónde está cada uno y cuál es su precio. Los objetos con precio positivo hay que pagarlos, suponiendo que se tenga suficiente dinero. Los de precio cero son regalos que hay que aceptar en cualquier caso. Los de precio negativo pueden tomarse o dejarse a voluntad del jugador.

4. Los enemigos con los que hay que luchar, así como la habilidad y la fuerza de cada uno, definida por un número de 1 a 20.

5. En algunos lugares (que llamaremos «condiciones») puede exigirse al programa que compruebe si el jugador lleva consigo determinado objeto y actúe de una u otra manera, según que la respuesta sea afirmativa o negativa. Estos lugares deben estar bien definidos.

6. Por último, una restricción: en una misma habitación no puede haber a la vez objetos y enemigos, objetos y condiciones o enemigos y condiciones.

Pueden existir habitaciones de cinco clases diferentes, a saber:

1. Puntos de bifurcación normales, donde el jugador debe elegir el camino que debe seguir entre varios posibles.

2. Lugares donde se encuentra un enemigo, con quien el jugador lucha. La pelea la realizará automáticamente el programa JUEGO, dando al jugador la posibilidad de escapar si lleva las de perder. Una pelea tiene, por tanto, tres resultados posibles: victoria del jugador, victoria del enemigo (en cuyo caso el juego termina) y abandono o huida decidida por el jugador. En cada uno de estos tres casos el juego continuará en una habitación que puede o no ser diferente. Véase, por ejemplo, el sitio número 5 en el ejemplo anterior, donde las tres alternativas envían al jugador a tres lugares distintos. Por el contrario, en el sitio número 10 dos de las opciones terminan en el mismo lugar.

3. Lugares donde pueden obtenerse objetos, tal como se explicó anteriormente.

4. Lugares donde se comprueba una condición, que consiste en asegurarse de que el jugador tiene en su poder cierto objeto determinado. Si lo tiene, continuará su camino en cierta dirección. Si no lo tiene, en otra diferente.

5. Puntos finales del juego. Cuando el jugador llega a ellos, el juego habrá terminado.

En el caso del programa HAZJUEGO, la variable CL contiene la clase de cada una de las «habitaciones», mientras que las variables V1, V2, V3 y V4 contienen las conexiones de unas habitaciones con otras, dependiendo de su clase.

Una vez que tenemos totalmente definido sobre el papel el juego que vamos a construir, podemos ejecutar el programa HAZJUEGO, que nos pedirá los datos correspondientes y construirá con ellos un fichero que contiene la estructura completa del juego, y que servirá de punto de partida para los otros dos programas de este grupo.

El programa HAZJUEGO consta de las siguientes partes:

- Las líneas 1 a 95, que definen las variables que va a usar el programa.
- Las líneas 100 a 230, que constituyen el bucle principal de definición de habitaciones. Primero se pide el texto correspondiente a la siguiente habitación (si no se da, el programa da por terminada la creación del juego). Después se pide su clase y, en función de ésta, se salta a la parte del programa correspondiente (instrucción 230).
- Las líneas 1000 a 1140 corresponden a la definición de un punto de bifurcación.
- Las líneas 2000 a 2080 corresponden a la definición de una habitación de lucha contra un enemigo.
- Las líneas 3000 a 3140 corresponden a la definición de un lugar donde pueden obtenerse objetos.
- Las líneas 4000 a 4130 corresponden a la definición de un lugar donde se comprueba una condición.
- Finalmente, las líneas 5000 a 5010 definen un punto final del juego.

• Los cinco casos anteriores se reúnen en las líneas 6000 a 6010, donde se cierra el bucle de creación de habitaciones y se regresa a la instrucción 100.

• Las líneas 9000 a 9390 se ejecutan cuando se ha dado por terminada la definición, y tienen por objeto guardar la definición del nuevo juego en un fichero en disco o en cinta (dependiendo del ordenador). Por último, la instrucción 9999 pone punto final a la ejecución del programa HAZJUEGO.

Veamos cómo se le proporcionarían a HAZJUEGO los datos del juego cuyo grafo hemos dado más arriba:

```
load "hazjuego
Ok
run
Sitio número 1
¿Qué te dicen?
? Empieza la aventura. Elige el camino
¿De qué clase es?
1: Elegir camino
2: Luchar
3: Obtener objetos
4: Comprobar objetos
5: Fin del juego
? 1
Caso 1
? Este
¿A dónde va? 2
Caso 2
? Sur
¿A dónde va? 3
Caso 3
?
Sitio número 2
¿Qué te dicen?
? Nueva bifurcación
¿De qué clase es?
1: Elegir camino
2: Luchar
3: Obtener objetos
4: Comprobar objetos
5: Fin del juego
? 1
Caso 1
? Sudoeste
```

¿A dónde va? 3
Caso 2
? Sur
¿A dónde va? 5
Caso 3
?
Sitio número 3
¿Qué te dicen?
? Estás en una tienda
¿De qué clase es?
1: Elegir camino
2: Luchar
3: Obtener objetos
4: Comprobar objetos
5: Fin del juego
? 3
Objeto 1
? Tijeras
¿Precio? 20
Objeto 2
? Papel
¿Precio? 10
Objeto 3
?
¿A dónde vas después? 4
Sitio número 4
¿Qué te dicen?
? El camino se divide en dos
¿De qué clase es?
1: Elegir camino
2: Luchar
3: Obtener objetos
4: Comprobar objetos
5: Fin del juego
? 1
Caso 1
? Nordeste
¿A dónde va? 2
Caso 2
? Este
¿A dónde va? 5
Caso 3
?
Sitio número 5

¿Qué te dicen?
? Prepárate a luchar con un duende
¿De qué clase es?
1: Elegir camino
2: Luchar
3: Obtener objetos
4: Comprobar objetos
5: Fin del juego
? 2
¿Habilidad del enemigo? 6
¿Fuerza del enemigo? 5
¿A dónde vas si ganas? 8
¿A dónde vas si abandonas? 7
¿A dónde vas si pierdes? 6
Sitio número 6
¿Qué te dicen?
? Lo siento. Has perdido
¿De qué clase es?
1: Elegir camino
2: Luchar
3: Obtener objetos
4: Comprobar objetos
5: Fin del juego
? 5
Sitio número 7
¿Qué te dicen?
? Tienes que cortar una cinta para pasar
¿De qué clase es?
1: Elegir camino
2: Luchar
3: Obtener objetos
4: Comprobar objetos
5: Fin del juego
? 4
¿Qué objeto tienes que tener?
? Tijeras
¿A dónde vas si lo tienes? 8
¿A dónde vas si no lo tienes? 6
Sitio número 8
¿Qué te dicen?
? Encuentras un tesoro
¿De qué clase es?
1: Elegir camino
2: Luchar

```
3: Obtener objetos
4: Comprobar objetos
5: Fin del juego
? 3
Objeto 1
? Diamante
¿Precio? -1
Objeto 2
?
¿A dónde vas después? 9
Sitio número 9
¿Qué te dicen?
? Un ogro no te deja pasar
¿De qué clase es?
1: Elegir camino
2: Luchar
3: Obtener objetos
4: Comprobar objetos
5: Fin del juego
? 2
¿Habilidad del enemigo? 12
¿Fuerza del enemigo? 10
¿A dónde vas si ganas? 10
¿A dónde vas si abandonas? 6
¿A dónde vas si pierdes? 6
Sitio número 10
¿Qué te dicen?
? Enhorabuena. Has ganado
¿De qué clase es?
1: Elegir camino
2: Luchar
3: Obtener objetos
4: Comprobar objetos
5: Fin del juego
? 5
Sitio número 11
¿Qué te dicen?
?
¿Nombre de la aventura? juego
Ok
```

Como se observará, existen habitaciones de cinco clases diferentes, a saber:

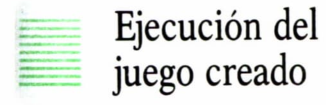
1. Puntos de bifurcación normales, donde el jugador debe elegir el camino que debe seguir entre varios posibles.

2. Lugares donde se encuentra un enemigo, con quien el jugador lucha. La pelea la realizará automáticamente el programa JUEGO, dando al jugador la posibilidad de escapar si lleva las de perder. Una pelea tiene, por tanto, tres resultados posibles: victoria del jugador, victoria del enemigo (en cuyo caso el juego termina) y abandono o huida decidida por el jugador. En cada uno de estos tres casos el juego continuará en una habitación que puede o no ser diferente. Véase, por ejemplo, el sitio número 5 en el ejemplo anterior, donde las tres alternativas envían al jugador a tres lugares distintos. Por el contrario, en el sitio número 10 dos de las opciones terminan en el mismo lugar.

3. Lugares donde pueden obtenerse objetos, tal como se explicó anteriormente.

4. Lugares donde se comprueba una condición, que consiste en asegurarse de que el jugador tiene en su poder cierto objeto determinado. Si lo tiene, continuará su camino en cierta dirección. Si no lo tiene, en otra diferente.

5. Puntos finales del juego. Cuando el jugador llega a ellos, el juego habrá terminado.



Ejecución del juego creado

A continuación vamos a ver el programa que permite poner en práctica cualquiera de los juegos creados anteriormente (es decir, jugar):

Programa 5.2. Programa JUEGO en lenguaje BASIC.

Este programa es válido para SPECTRUM, AMSTRAD, COMMODORE e IBM PC o compatibles (ver notas).

```
1   REM Ejecución de un librojuego. Por M. Alfonseca.
10  DIM S$(300): DIM E$(800): DIM O$(100)
15  DIM CL(300): DIM IN(300)
20  DIM E1(300): DIM O1(300):
25  DIM V1(800): DIM V2(250): DIM V3(100): DIM V4(170)
30  DIM HB(100): DIM FZ(100): DIM PR(100): DIM CN(100)
100 REM Carga del juego desde disco o cinta
110 INPUT "¿Nombre de la aventura";F$
120 IF F$="" THEN GOTO 9999
140 OPEN F$ FOR INPUT AS #1
150 INPUT #1,I,IE,IE1,IM,IO,IO1,IC
160 FOR K=1 TO I
170 INPUT #1,S$(K)
180 NEXT K
```

```
190  FOR K=1 TO I
200  INPUT #1,CL(K),IN(K)
210  NEXT K
250  FOR K=1 TO IE
260  INPUT #1,E$(K)
270  NEXT K
280  FOR K=1 TO IE
290  INPUT #1,V1(K)
300  NEXT K
310  FOR K=1 TO IE1
320  INPUT #1,E1(K)
330  NEXT K
340  FOR K=1 TO IM
350  INPUT #1,HB(K),FZ(K),V2(3*K-2),V2(3*K-1),V2(3*K)
360  NEXT K
370  FOR K=1 TO IO
380  INPUT #1,O$(K)
390  NEXT K
400  FOR K=1 TO IO
410  INPUT #1,PR(K)
420  NEXT K
430  FOR K=1 TO IO1
440  INPUT #1,O1(K),V3(K)
450  NEXT K
460  FOR K=1 TO IC
470  INPUT #1,CN(K),V4(2*K-1),V4(2*K)
480  NEXT K
490  CLOSE #1
999  REM Comienza el juego
1000 LET PO=1
1010 DIM E2(401): DIM O2(401): DIM TI(100): DIM NO(30)
1020 FOR J=1 TO IE1
1030 FOR K=1 TO J
1040 LET E2(J+1)=E2(J)+E1(K)
1050 NEXT K
1060 NEXT J
1070 FOR J=1 TO IO1
1080 FOR K=1 TO J
1090 LET O2(J+1)=O2(J)+O1(K)
1100 NEXT K
1110 NEXT J
1120 LET IT=0: REM Indice de cosas que tienes
1130 RANDOMIZE
1140 LET HA=10+INT(RND*10)
1150 LET FU=10+INT(RND*10)
1160 LET DO=200+INT(RND*100)
1170 PRINT "Tu habilidad es ";HA
1180 PRINT "Tu fuerza es ";FU
1190 PRINT "Tienes ";DO;" pesetas"
2000 REM Bucle principal de juego
```

```
2010 PRINT S$(PO)
2020 LET A=IN(PO)
2030 ON CL(PO) GOTO 3000,4000,5000,6000,7000
3000 REM Elegir camino
3010 LET B=1
3020 IF E1(A)=1 THEN GOTO 3070
3030 FOR K=1 TO E1(A)
3040 PRINT K;":";E$(E2(A)+K)
3050 NEXT K
3060 INPUT B
3070 LET PO=V1(E2(A)+B)
3080 GOTO 2000
4000 REM Lucha contra un enemigo
4010 LET H=HB(A)
4020 LET F=FZ(A)
4030 IF FU>4 THEN GOTO 4080
4040 PRINT "Te quedan menos de cinco puntos de fuerza"
4050 PRINT "¿Quieres abandonar?"
4060 INPUT A$
4070 IF (A$="si") OR (A$="SI") THEN GOTO 4250
4080 LET AA=HA+1+INT(RND*11)
4090 LET BB=H+1+INT(RND*11)
4100 IF AA=BB THEN PRINT "Los dos fallasteis el golpe"
4110 IF AA>BB THEN PRINT "Le has dado un buen golpe"
4120 IF AA<BB THEN PRINT "Te ha dado un buen golpe. Pierdes un punto
de fuerza"
4130 IF AA<BB THEN FU=FU-1
4135 FOR V=1 TO 200: NEXT V: REM Pausa
4140 IF FU=0 THEN GOTO 4220
4150 IF AA>BB THEN F=F-1
4160 IF F=0 THEN GOTO 4190
4180 GOTO 4030: REM Sigue la lucha
4190 PRINT "Le has matado. Enhorabuena"
4200 LET PO=V2(3*A-2)
4210 GOTO 2000
4220 PRINT "Te ha vencido. Lo siento"
4230 LET PO=V2(3*A)
4240 GOTO 2000
4250 LET PO=V2(3*A-1): REM Abandono
4260 GOTO 2000
5000 REM Obtener objetos
5005 LET Q=0
5010 FOR K=1 TO O1(A)
5020 IF PR(O2(A)+K)<>0 THEN GOTO 5080
5030 IF Q=0 THEN PRINT "Te dan los siguientes objetos:"
5040 LET Q=1
5050 PRINT O$(O2(A)+K)
5060 LET TI(IT+1)=O2(A)+K
5070 LET IT=IT+1
5080 NEXT K
```

```
5085 LET Q=0
5090 FOR K=1 TO O1(A)
5100 IF PR(O2(A)+K)<=0 THEN GOTO 5150
5110 IF Q=0 THEN PRINT "Puedes comprar los siguientes objetos:"
5115 IF Q=0 THEN PRINT "Tienes ";DO;" pesetas"
5120 LET Q=Q+1
5130 PRINT Q;":";O$(O2(A)+K)
5140 LET NO(Q)=O2(A)+K
5150 NEXT K
5155 IF Q=0 THEN GOTO 5290
5160 PRINT "¿Cuántos quieres?"
5170 INPUT B
5180 IF B=0 GOTO 5285
5190 PRINT "¿Qué números quieres (dámelos de uno en uno)?"
5200 FOR K=1 TO B
5210 INPUT N
5220 LET N=NO(N)
5230 IF PR(N)>DO THEN GOTO 5270
5240 LET TI(IT+1)=N
5245 LET IT=IT+1
5250 LET DO=DO-PR(N)
5260 GOTO 5280
5270 PRINT "No tienes bastante dinero para comprar ";O$(N)
5280 NEXT K
5285 LET Q=0
5290 FOR K=1 TO O1(A)
5300 IF PR(O2(A)+K)>=0 THEN GOTO 5350
5310 IF Q=0 THEN PRINT "Puedes coger los siguientes objetos:"
5320 LET Q=Q+1
5330 PRINT Q;":";O$(O2(A)+K)
5340 LET NO(Q)=O2(A)+K
5350 NEXT K
5355 IF Q=0 THEN GOTO 5470
5360 PRINT "¿Cuántos quieres?"
5370 INPUT B
5380 IF B=0 GOTO 5470
5390 PRINT "¿Qué números quieres?"
5400 FOR K=1 TO B
5410 INPUT N
5420 LET N=NO(N)
5440 LET TI(IT+1)=N
5450 LET IT=IT+1
5460 NEXT K
5470 LET PO=V3(A)
5480 GOTO 2000
6000 REM Comprobar objetos
6005 FOR K=1 TO IT
6010 IF CN(A)=TI(K) THEN GOTO 6060
6020 NEXT K
6030 PRINT "Lo siento. No tienes ";O$(CN(A))
```

```
6040 LET PO=V4(2*A)
6050 GOTO 2000
6060 PRINT "Tienes suerte. Tenías ";O$(CN(A))
6070 LET PO=V4(2*A-1)
6080 GOTO 2000
7000 REM Casilla final del juego
9999 END
```

NOTAS: El programa anterior es válido directamente para IBM PC.

Para SPECTRUM cambiar las siguientes líneas:

```
10   DIM S$(300,15): DIM E$(800,8): DIM O$(100,8)
2030 GOTO 2000+1000*CL(PO)
9999 STOP
```

Sustituir también las líneas comprendidas entre 140 y 490 por las siguientes:

```
140  DIM A(7)
150  LET I=A(1)
160  LET IE=A(2)
170  LET IL=A(3)
180  LET IM=A(4)
190  LET IO=A(5)
200  LET IJ=A(6)
210  LET IC=A(6)
220  LOAD F$ DATA A()
230  LOAD F$ DATA S$()
240  LOAD F$ DATA E$()
250  LOAD F$ DATA O$()
260  LOAD F$ DATA CL()
270  LOAD F$ DATA IN()
280  LOAD F$ DATA E1()
290  LOAD F$ DATA O1()
300  LOAD F$ DATA V1()
310  LOAD F$ DATA V2()
320  LOAD F$ DATA V3()
330  LOAD F$ DATA V4()
340  LOAD F$ DATA HB()
350  LOAD F$ DATA FZ()
360  LOAD F$ DATA PR()
370  LOAD F$ DATA CN()
```

Para AMSTRAD cambiar las siguientes líneas:

```
140  OPENIN F$
490  CLOSEIN
1140 LET HA=10+INT(RND(1)*10)
1150 LET FU=10+INT(RND(1)*10)
```

```
1160 LET DO=200+INT(RND(1)*100)
4080 LET AA=HA+1+INT(RND(1)*11)
4090 LET BB=H+1+INT(RND(1)*11)
```

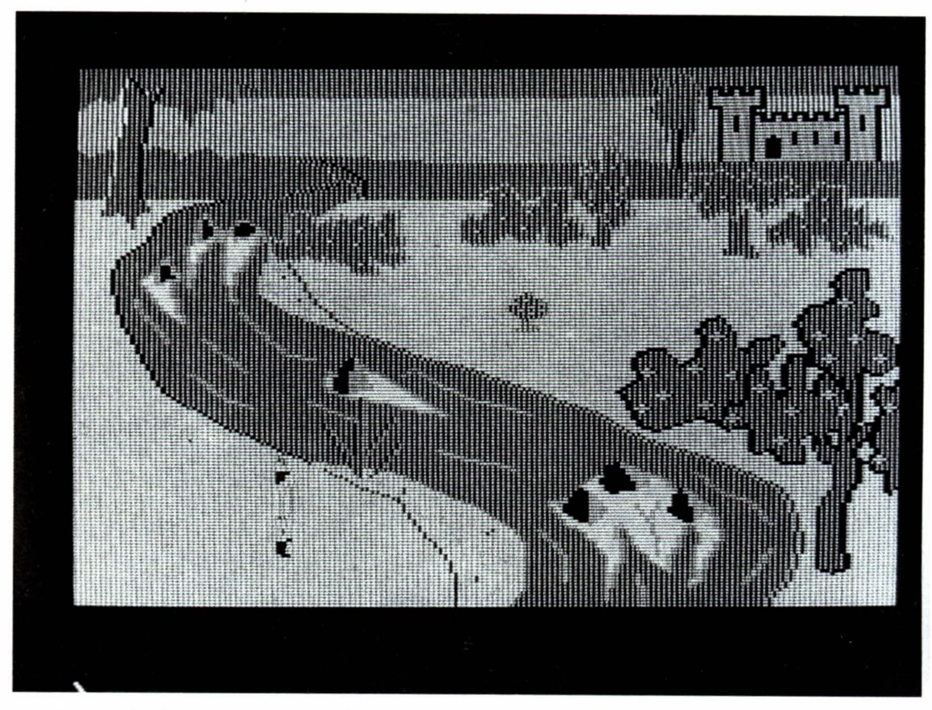
y cambiar todas las referencias a #1 por #9.

Para COMMODORE cambiar las siguientes líneas:

```
140  OPEN 1,1,0,F$
490  CLOSE 1
1130 LET Z=RND(-TI)
1140 LET HA=10+INT(RND(1)*10)
1150 LET FU=10+INT(RND(1)*10)
1160 LET DO=200+INT(RND(1)*100)
4080 LET AA=HA+1+INT(RND(1)*11)
4090 LET BB=H+1+INT(RND(1)*11)
```

Programa 5.2. *Programa JUEGO*

Este programa comienza por leer de un disco o cinta el fichero creado por HAZJUEGO, que contiene la definición de un juego de búsqueda de te-



Ríos, árboles, rocas y castillos son elementos ordinarios en la topografía de «King's Quest».

soros. El programa calcula aleatoriamente la habilidad y la fuerza del jugador así como el dinero de que dispone. (Recuérdese que la instrucción 1130 puede sustituirse por RANDOMIZE en las máquinas cuyo intérprete de BASIC no la acepte en la forma que indica el programa.) A continuación, el programa coloca al jugador en la «habitación» número 1 y, en función de la clase de ésta, permite al jugador elegir el camino a seguir, o le hace enfrentarse con un enemigo, o le permite comprar, recibir o recoger objetos, o comprueba si tiene en su poder un objeto preestablecido. El mismo proceso se repite en cada habitación sucesiva por donde va pasando el jugador, hasta que éste llegue a un punto final, en cuyo momento el programa se detiene.

Veamos un ejemplo de la ejecución del programa JUEGO con el juego definido en el apartado anterior:

```
load "juego
Ok
run
¿Nombre de la aventura? juego
Tu habilidad es 12
Tu fuerza es 11
Tienes 295 pesetas
Empieza la aventura. Elige el camino
 1 : Este
 2 : Sur
? 1
Nueva bifurcación
 1 : Sudoeste
 2 : Sur
? 1
Estás en una tienda
Puedes comprar los siguientes objetos:
Tienes 295 pesetas
 1 : Tijeras
 2 : Papel
¿Cuántos quieres?
? 1
¿Qué números quieres (dámelos de uno en uno)?
? 1
El camino se divide en dos
 1 : Nordeste
 2 : Este
? 2
Prepárate a luchar con un duende
Le has dado un buen golpe
```

Le has dado un buen golpe
Le has dado un buen golpe
Le has dado un buen golpe
Le has dado un buen golpe
Le has matado. Enhorabuena
Encuentras un tesoro
Puedes coger los siguientes objetos:
1 : Diamante
¿Cuántos quieres?
? 1
¿Qué números quieres?
? 1
Un ogro no te deja pasar
Le has dado un buen golpe
Te ha dado un buen golpe. Pierdes un punto de fuerza
Le has dado un buen golpe
Te ha dado un buen golpe. Pierdes un punto de fuerza
Te ha dado un buen golpe. Pierdes un punto de fuerza
Le has dado un buen golpe
Le has dado un buen golpe
Te ha dado un buen golpe. Pierdes un punto de fuerza
Te ha dado un buen golpe. Pierdes un punto de fuerza
Le has dado un buen golpe
Le has dado un buen golpe
Los dos fallasteis el golpe
Te ha dado un buen golpe. Pierdes un punto de fuerza
Le has dado un buen golpe
Te ha dado un buen golpe. Pierdes un punto de fuerza
Te quedan menos de cinco puntos de fuerza
¿Quieres abandonar?
? no
Te ha dado un buen golpe. Pierdes un punto de fuerza
Te quedan menos de cinco puntos de fuerza
¿Quieres abandonar?
? no
Le has dado un buen golpe
Te quedan menos de cinco puntos de fuerza
¿Quieres abandonar?
? no
Le has dado un buen golpe
Te quedan menos de cinco puntos de fuerza
¿Quieres abandonar?
? no
Le has dado un buen golpe

```
Le has matado. Enhorabuena
Enhorabuena. Has ganado
Ok
```



Descripción de un juego preestablecido

El tercer programa del grupo (llamado VERJUEGO) permite analizar un juego construido previamente por HAZJUEGO y presentarlo en una forma semejante a la de un libro-juego. Veamos el listado en BASIC de este programa:

Programa 5.3. Programa VERJUEGO en lenguaje BASIC.

Este programa es válido para SPECTRUM, AMSTRAD, COMMODORE e IBM PC o compatibles (ver notas).

```
1   REM Listado de un librojuego. Por M. Alfonseca.
10  DIM S$(300): DIM E$(800): DIM O$(100)
15  DIM CL(300): DIM IN(300)
20  DIM E1(300): DIM O1(300):
25  DIM V1(800): DIM V2(250): DIM V3(100): DIM V4(170)
30  DIM HB(100): DIM FZ(100): DIM PR(100): DIM CN(100)
100 REM Carga del juego desde disco o cinta
110 INPUT "¿Nombre de la aventura";F$
120 IF F$="" THEN GOTO 9999
140 OPEN F$ FOR INPUT AS #1
150 INPUT #1,I,IE,IE1,IM,IO,IO1,IC
160 FOR K=1 TO I
170 INPUT #1,S$(K)
180 NEXT K
190 FOR K=1 TO I
200 INPUT #1,CL(K),IN(K)
210 NEXT K
250 FOR K=1 TO IE
260 INPUT #1,E$(K)
270 NEXT K
280 FOR K=1 TO IE
290 INPUT #1,V1(K)
300 NEXT K
310 FOR K=1 TO IE1
320 INPUT #1,E1(K)
330 NEXT K
340 FOR K=1 TO IM
350 INPUT #1,HB(K),FZ(K),V2(3*K-2),V2(3*K-1),V2(3*K)
360 NEXT K
370 FOR K=1 TO IO
```

```
380 INPUT #1,O$(K)
390 NEXT K
400 FOR K=1 TO IO
410 INPUT #1,PR(K)
420 NEXT K
430 FOR K=1 TO IO1
440 INPUT #1,O1(K),V3(K)
450 NEXT K
460 FOR K=1 TO IC
470 INPUT #1,CN(K),V4(2*K-1),V4(2*K)
480 NEXT K
490 CLOSE #1
999 'Comienza el juego
1000 LET PO=1
1010 DIM E2(401),O2(401)
1020 FOR J=1 TO IE1
1030 FOR K=1 TO J
1040 LET E2(J+1)=E2(J)+E1(K)
1050 NEXT K
1060 NEXT J
1070 FOR J=1 TO IO1
1080 FOR K=1 TO J
1090 LET O2(J+1)=O2(J)+O1(K)
1100 NEXT K
1110 NEXT J
2000 'Bucle principal de juego
2005 IF S$(PO)="" THEN GOTO 9999
2010 PRINT PO;": ";S$(PO)
2020 LET A=IN(PO)
2030 ON CL(PO) GOTO 3000,4000,5000,6000,7000
3000 'Elegir camino
3010 LET B=1
3020 IF E1(A)=1 THEN GOTO 3070
3030 FOR K=1 TO E1(A)
3040 PRINT "  ";K;":";E$(E2(A)+K);" (";V1(E2(A)+K);")"
3050 NEXT K
3060 GOTO 8000
3070 PRINT "  Vas a ";E$(E2(A)+1);" (";V1(E2(A)+K);")"
3080 GOTO 8000
4000 'Lucha contra un enemigo
4010 PRINT "  Habilidad: ";HB(A);". Fuerza: ";FZ(A)
4020 PRINT "  Si ganas vas a ";V2(3*A-2)
4030 PRINT "  Si abandonas vas a ";V2(3*A-1)
4040 PRINT "  Si pierdes vas a ";V2(3*A)
4050 GOTO 8000
5000 'Obtener objetos
5005 LET Q=0
5010 FOR K=1 TO O1(A)
5020 IF PR(O2(A)+K)<>0 THEN GOTO 5080
5030 IF Q=0 THEN PRINT "  Te dan los siguientes objetos:"
```

```
5040 LET Q=1
5050 PRINT "    ";O$(O2(A)+K)
5080 NEXT K
5085 LET Q=0
5090 FOR K=1 TO O1(A)
5100 IF PR(O2(A)+K)<=0 THEN GOTO 5150
5110 IF Q=0 THEN PRINT "  Puedes comprar los siguientes objetos:"
5120 LET Q=Q+1
5130 PRINT "  ";Q;":";O$(O2(A)+K);" (";PR(O2(A)+K);")"
5150 NEXT K
5285 LET Q=0
5290 FOR K=1 TO O1(A)
5300 IF PR(O2(A)+K)>=0 THEN GOTO 5350
5310 IF Q=0 THEN PRINT "  Puedes coger los siguientes objetos:"
5320 LET Q=Q+1
5330 PRINT "  ";Q;":";O$(O2(A)+K)
5350 NEXT K
5360 PRINT "  Después vas a ";V3(A)
5480 GOTO 8000
6000 'Comprobar objetos
6010 PRINT "  Comprueba si tienes ";O$(CN(A))
6020 PRINT "  Sí: Vas a ";V4(2*A-1)
6030 PRINT "  No: Vas a ";V4(2*A)
6040 GOTO 8000
7000 'Casilla final del juego
7010 PRINT "  Fin del juego"
8000 LET PO=PO+1
8010 GOTO 2000
9999 END
```

NOTAS: El programa anterior es válido directamente para IBM PC.

Para SPECTRUM cambiar las siguientes líneas:

```
10   DIM S$(300,15): DIM E$(800,8): DIM O$(100,8)
2030 GOTO 2000+1000*CL(PO)
9999 STOP
```

Sustituir también las líneas comprendidas entre 140 y 490 por las siguientes:

```
140  DIM A(7)
150  LET I=A(1)
160  LET IE=A(2)
170  LET IL=A(3)
180  LET IM=A(4)
190  LET IO=A(5)
200  LET IJ=A(6)
210  LET IC=A(6)
220  LOAD F$ DATA A()
230  LOAD F$ DATA S$()
```

```
240  LOAD F$ DATA E$()
250  LOAD F$ DATA O$()
260  LOAD F$ DATA CL()
270  LOAD F$ DATA IN()
280  LOAD F$ DATA E1()
290  LOAD F$ DATA O1()
300  LOAD F$ DATA V1()
310  LOAD F$ DATA V2()
320  LOAD F$ DATA V3()
330  LOAD F$ DATA V4()
340  LOAD F$ DATA HB()
350  LOAD F$ DATA FZ()
360  LOAD F$ DATA PR()
370  LOAD F$ DATA CN()

Para AMSTRAD cambiar las siguientes líneas:

140  OPENIN F$
490  CLOSEIN

y cambiar todas las referencias a #1 por #9.

Para COMMODORE cambiar las siguientes líneas:

140  OPEN 1,1,0,F$
490  CLOSE 1
```

Programa 5.3. *Programa VERJUEGO*

Veamos ahora el resultado de su ejecución con el fichero definido anteriormente:

```
load "verjuego
Ok
run
¿Nombre de la aventura? juego
 1 : Empieza la aventura. Elige el camino
   1 : Este ( 2 )
   2 : Sur ( 3 )
 2 : Nueva bifurcación
   1 : Sudoeste ( 3 )
   2 : Sur ( 5 )
 3 : Estás en una tienda
  Puedes comprar los siguientes objetos:
   1 : Tijeras ( 20 )
   2 : Papel ( 10 )
```

```
4 : El camino se divide en dos
  1 : Nordeste ( 2 )
  2 : Este ( 5 )
5 : Prepárate a luchar con un duende
 Habilidad: 6 . Fuerza: 5
 Si ganas, vas a 8
 Si abandonas, vas a 7
 Si pierdes, vas a 6
6 : Lo siento. Has perdido
 Fin del juego
7 : Tienes que cortar una cinta para pasar
 Comprueba si tienes Tijeras
 Sí: Vas a 8
 No: Vas a 6
8 : Encuentras un tesoro
 Puedes coger los siguientes objetos:
  1 : Diamante
9 : Un ogro no te deja pasar
 Habilidad: 12 . Fuerza: 10
 Si ganas, vas a 10
 Si abandonas, vas a 6
 Si pierdes, vas a 6
10 : Enhorabuena. Has ganado
 Fin del juego
Ok
```

Se observará que los números de las habitaciones corresponden a las páginas de un libro-juego típico. Si se desea complicarlo, puede ahora sustituirse la línea que explica lo que ocurre en cada habitación por un texto más o menos largo (quizá una página). Naturalmente, este ejemplo es muy sencillo, pues sólo tiene 10 lugares distintos, es decir, daría lugar a un libro-juego corto de 10 páginas, pero los tres programas dados más arriba permiten complicar el juego todo lo que se quiera. No hay nada que impida, en efecto, construir un libro-juego de 400 páginas utilizando este método.

Si se desea obtener el libro-juego en una impresora, pueden utilizarse dos métodos distintos:

- Algunas máquinas permiten obtener automáticamente una copia impresa de todo lo que aparezca en la pantalla, sin más que apretar una tecla determinada. Si este es el caso, basta con presionar dicha tecla antes de ejecutar el programa VERJUEGO para obtener el libro-juego en papel.
- En caso de que no exista tal posibilidad, pueden sustituirse todas las instrucciones PRINT del programa VERJUEGO por la instrucción LPRINT.

En este caso, el libro-juego quedará impreso sin que se escriba nada en la pantalla.

Los tres juegos anteriores están escritos en el lenguaje BASIC, que es uno de los más extendidos en la actualidad, porque su pobreza sintáctica hace relativamente fácil construir intérpretes muy pequeños, lo que tenía gran importancia en los microordenadores primitivos, que disponían de poca memoria. Sin embargo, el gran abaratamiento de las memorias de ordenador y la progresiva proliferación de ordenadores más potentes harán perder terreno al BASIC, dando paso a lenguajes más completos como PASCAL o APL. De hecho, la mayor parte de los programas descritos en este libro fueron construidos primero en APL, aunque después se los tradujo al BASIC por las razones antes indicadas.

APENDICE 1

El programa que sigue es una versión para Spectrum, del programa «La guerra espacial», que figura en el capítulo 3, en versión IBM PC:

```
  10 REM Star trek. Por M. Alfonseca.
  20 LET X=0: LET XH=0: LET XK=0: CLS : PRINT "Espera un momento. Estoy creando
la Galaxia."
  30 DIM G(8,8)
  40 DIM Q$(8,32): REM Mapa del cuadrante de la Galaxia"
  50 DIM Z(10,10): REM Mapa acumulado de la Galaxia"
  60 DIM F(8): REM Danos en la nave AVENTURA
  70 DIM K(10,3): REM Naves enemigas en este cuadrante
  80 DIM I(1,2): REM Bases estelares en este cuadrante
  90 DIM E(8,2): REM Estrellas en este cuadrante
  99 REM Variables C utilizadas por TORPEDO
 100 DIM A(8): DIM B(8): DIM C(8): DIM D(8)
 110 LET A(1)=0: LET A(2)=-1: LET A(3)=1: LET A(4)=-1: LET A(5)=0: LET A(6)=1: L
ET A(7)=1: LET A(8)=1
 120 LET B(1)=-1: LET B(2)=0: LET B(3)=0: LET B(4)=1: LET B(5)=1: LET B(6)=0: LE
T B(7)=0: LET B(8)=-1
 130 LET C(1)=1: LET C(2)=1: LET C(3)=0: LET C(4)=-1: LET C(5)=-1: LET C(6)=-1:
LET C(7)=0: LET C(8)=1
 140 LET D(1)=0: LET D(2)=-1: LET D(3)=-1: LET D(4)=0: LET D(5)=0: LET D(6)=1: L
ET D(7)=1: LET D(8)=0
1000 RANDOMIZE : REM Comienza la creacion de una aventura espacial
1010 LET B9=0: LET K9=0: REM Generacion de estrellas, bases y naves enemigas
1020 FOR I=1 TO 8: FOR J=1 TO 8: LET A=RND
1030 LET AK=0
1040 IF A>.8 THEN LET AK=1
1050 IF A>.95 THEN LET AK=2
1060 IF A>.9799999 THEN LET AK=3
1070 LET K9=K9+AK
1080 LET A=RND
1090 LET AB=0
1100 IF A>.96 THEN LET AB=1
1110 LET B9=B9+AB
1120 LET AS=1+INT (8*RND)
1130 LET G(I,J)=AS+10*AB+100*AK
1140 NEXT J: NEXT I
1150 LET K7=K9
1200 REM Posicion de la nave AVENTURA
1210 LET Q1=1+INT (8*RND): LET Q2=1+INT (8*RND)
1220 LET S1=1+INT (8*RND): LET S2=1+INT (8*RND)
1230 REM Fechas
1240 LET T0=100*(20+INT (20*RND)): REM Fecha inicial
1250 LET T=T0: REM Fecha actual
1260 LET T9=30: IF K9>20 THEN LET T9=30: REM Fecha final
```

```
1270 REM Otros datos
1280 LET E0=3000: REM Energia inicial
1290 LET E1=E0: REM Energia actual
1300 LET P0=10: REM Numero inicial de torpedos
1310 LET P1=P0: REM Numero actual de torpedos
1320 LET S0=0: REM escudo protector actual
1500 REM Comienza la aventura
1510 CLS : PRINT "Al capitan de la nave AVENTURA. Estas son las ordenes:"
1520 PRINT "Debe usted destruir las ";K9;".naves enemigas que han invadido la Ga
"
1530 PRINT "laxia, antes de que puedan ata-"
1540 PRINT "car el cuartel general de la Fe-deracion en la fecha estelar "
1550 PRINT T0-T9;". Esto le deja ";T9;" fechas."
1560 PRINT "Hay ";B9;" bases estelares en la Ga- laxia, donde puede reabastecer"
1570 PRINT "la nave AVENTURA. Buena suerte!"
1580 PRINT : INPUT "Presione ENTER cuando este dispuesto a asumir el mando.";A$
1600 CLS : GO SUB 1610: GO TO 2000: REM Inicializar pantalla
1610 PRINT AT 0,0;"P. corto alcance"
1650 PRINT AT 13,16;"FECHA"
1680 PRINT AT 14,16;"CONDICION"
1700 PRINT AT 15,16;"CUADRANTE"
1720 PRINT AT 16,16;"SECTOR"
1740 PRINT AT 17,16;"ENERGIA"
1760 PRINT AT 18,16;"TORPEDOS"
1780 PRINT AT 19,16;"ESCUDO"
1790 RETURN
2000 REM Sala de control
2010 GO SUB 2020: GO TO 2200
2020 GO SUB 1610: PRINT AT 13,0;"P.largo alcante"
2030 IF F(3)<0 THEN GO TO 2150
2040 PRINT "_______________"
2050 FOR I=1 TO 3: PRINT AT 14+I,0;: FOR J=1 TO 3
2060 LET A=0
2065 LET A1=Q1+I: LET A2=Q2+J
2070 IF (A1>2) AND (A1<11) AND (A2>2) AND (A2<11) THEN LET A=G(A1-2,A2-2)
2080 LET Z(Q1+I-1,Q2+J-1)=A: PRINT ("    "+STR$ (A))(1+(A>9)+(A>99) TO 4+(A>9)+(A
>99));
2090 NEXT J: NEXT I
2100 PRINT AT 18,0;"________________": RETURN
2150 GO SUB 9947: PRINT : PRINT "P. LARGO ALCANCE ESTROPEADA"
2160 FOR I=1 TO 5: PRINT AT 13+I,0;: FOR J=0 TO 15
2170 PRINT " ";
2180 NEXT J: NEXT I: RETURN
2200 REM Entramos en un nuevo cuadrante"
2210 PRINT AT 15,26;Q1;",";Q2
2230 LET S3=G(Q1,Q2): LET K3=INT (S3/100): LET S3=S3-100*K3: LET B3=INT (S3/10):
 LET S3=S3-10*B3
2240 REM Distribucion en el cuadrante
2250 FOR I=1 TO 8: FOR J=1 TO 32: LET Q$(I,J)=" ": NEXT J: NEXT I
2260 LET Q$(S1,1+4*(S2-1))="<"
2270 LET Q$(S1,2+4*(S2-1))="="
2280 LET Q$(S1,3+4*(S2-1))=">"
2290 LET V$=".": LET W$=".": LET X$="."
2300 FOR I=1 TO K3: GO SUB 2400: LET K(I,1)=R1: LET K(I,2)=R2: LET K(I,3)=200: N
EXT I
2303 IF K3=3 THEN GO TO 2310
2305 FOR I=K3+1 TO 3: LET K(I,3)=0: NEXT I
2310 LET V$="-": LET W$="O": LET X$="-"
2320 FOR I=1 TO B3: GO SUB 2400: LET I(I,1)=R1: LET I(I,2)=R2: NEXT I
2330 LET V$=" ": LET W$="*": LET X$=" "
2340 FOR I=1 TO S3: GO SUB 2400: LET E(I,1)=R1: LET E(I,2)=R2: NEXT I
2350 GO TO 2500
2400 LET R1=1+INT (8*RND): LET R2=1+INT (8*RND)
2410 IF Q$(R1,2+4*(R2-1))<>" " THEN GO TO 2400
2420 LET Q$(R1,1+4*(R2-1))=V$
2430 LET Q$(R1,2+4*(R2-1))=W$
2440 LET Q$(R1,3+4*(R2-1))=X$
2450 RETURN
2500 REM Comprobacion de posicion de amarre con estacion
2510 IF B3=0 THEN GO TO 2600: REM No hay base
2520 IF S2<>I(1,2) THEN GO TO 2600: REM No estamos amarrados
```

```
2530 IF (S1<I(1,1)-1) OR (S1>I(1,1)+1) THEN GO TO 2600
2540 PRINT AT 14,26;"PUERTO"
2550 LET D0=1: REM En puerto
2560 LET E1=E0: LET P1=P0: LET S0=0: REM Repostamos
2570 GO SUB 9947: PRINT "Sin escudo protector para el amarre"
2580 GO TO 2750
2600 LET D0=0: REM No estamos amarrados
2610 IF K3>0 THEN GO TO 2650
2620 IF E1<.1*E0 THEN GO TO 2700
2630 PRINT AT 14,26;"NORMAL"
2640 GO TO 2750
2650 PRINT AT 14,26;"ROJA  "
2660 BEEP 1,15
2670 IF S0>200 THEN GO TO 2750
2680 GO SUB 9947: PRINT : PRINT "    Escudo protector muy bajo"
2690 GO TO 2750
2700 PRINT AT 14,26;"AMBAR "
2750 IF F(2)<0 THEN GO TO 2800: REM Pantalla de corto alcance
2760 PRINT AT 1,0;"________________________________"
2770 FOR I=1 TO 8: PRINT AT 1+I,0;
2780 FOR J=1 TO 32: PRINT Q$(I,J);;: NEXT J: NEXT I
2790 GO TO 2820
2800 GO SUB 9947: PRINT "P. corto alcance estropeada"
2810 FOR I=1 TO 8: PRINT AT 1+I,0;: FOR J=1 TO 32: PRINT " ";: NEXT J: NEXT I
2820 PRINT AT 12,0;"________________________________"
2830 PRINT AT 13,26;T;" "
2840 PRINT AT 16,26;S1;",";S2;"  "
2850 PRINT AT 17,26;E1+S0;" "
2860 PRINT AT 18,26;P1;"  "
2870 PRINT AT 19,26;S0;"  "
2900 REM Comprobacion de combustible
2910 IF S0+E1<10 THEN GO TO 2930
2920 IF (E1>10) OR (F(7)=0) THEN GO TO 3000
2930 BEEP 1,15
2940 CLS : PRINT "ERROR FATAL!!! La nave ha quedado varada en el espacio"
2950 PRINT "No hay energia suficiente, y el control de escudo no puede"
2960 PRINT "pasar energia a la sala se maquinas."
2970 GO TO 9994
2999 REM Bucle de peticion de ordenes
3000 GO SUB 9947: PRINT AT 20,12;"ORDENES       "
3030 PRINT #0;AT 0,0;"Q:NAV W:CAN E:TOR R:ESC T:DAN   Y:COM U:FIN";
3040 GO SUB 9997: REM Lee una tecla
3060 IF I=CODE "U" THEN CLS : GO TO 9994: REM Fin
3080 IF I=CODE "Q" THEN GO TO 4000
3090 IF I=CODE "W" THEN GO TO 5000
3100 IF I=CODE "E" THEN GO TO 6000
3110 IF I=CODE "R" THEN GO TO 7000
3120 IF I=CODE "T" THEN GO TO 8000
3130 IF I=CODE "Y" THEN GO TO 9000
3140 GO TO 3040
4000 REM Control de navegacion
4010 INPUT "CURSO (1-9):  ";C1: IF C1<1 OR C1>9 THEN GO TO 4010
4030 INPUT "VELOCIDAD (0-8): ";W1: IF W1<0 OR W1>8 THEN GO TO 4030
4050 IF C1=9 THEN LET C1=1
4060 IF (W1<=.2) OR (F(1)>=0) THEN GO TO 4100
4070 GO SUB 9947: PRINT "MOTORES DANADOS.";#0;AT 0,0;"VELOCIDAD MAXIMA=0.2": GO
SUB 9950
4090 GO TO 4000
4100 LET N=INT (.5+8*W1): IF E1>=N THEN GO TO 4150
4110 GO SUB 9947: PRINT "ENERGIA INSUFICIENTE PARA MANIOBRAR": GO SUB 9950
4120 IF (F(7)<0) OR (S0<N-E1) THEN GO TO 3000
4130 GO SUB 9947: PRINT S0;" UNIDADES DISPONIBLES PARA ESCUDO": GO SUB 9950
4140 GO TO 3000
4150 REM Comienza el movimiento
4160 REM Primero el enemigo ataca
4170 GO SUB 9915
4180 REM Reparaciones en marcha
4190 GO SUB 9932
4200 LET Z1=S1: LET Z2=S2
4210 LET Q$(INT (S1),1+4*(INT (S2)-1))=" "
4220 LET Q$(INT (S1),2+4*(INT (S2)-1))=" "
```

```
4230 LET Q$(INT (S1),3+4*(INT (S2)-1))=" "
4240 LET X1=A(INT (C1))+B(INT (C1))*(C1-INT (C1))
4250 LET X2=C(INT (C1))+D(INT (C1))*(C1-INT (C1))
4260 LET I=1
4270 LET S1=S1+X1: LET S2=S2+X2
4280 IF (S1<1) OR (S1>=9) OR (S2<1) OR (S2>=9) THEN GO TO 4400
4290 IF Q$(INT (S1),2+4*(INT (S2)-1))=" " THEN GO TO 4340
4300 LET S1=S1-X1: LET S2=S2-X2
4310 GO SUB 9947: PRINT "PARADA AUTOMATICA DE MOTORES DEBIDO A MALA NAVEGACION"
4330 GO TO 4350
4340 LET I=I+1: IF N>=I THEN GO TO 4270
4350 LET Q$(INT (S1),1+4*(INT (S2)-1))="<"
4360 LET Q$(INT (S1),2+4*(INT (S2)-1))="="
4370 LET Q$(INT (S1),3+4*(INT (S2)-1))=">"
4380 GO SUB 4900
4390 LET S1=INT (.5+S1): LET S2=INT (.5+S2): GO TO 2500
4400 LET Z1=(8*Q1)+Z1+N*X1
4410 LET Z2=(8*Q2)+Z2+N*X2
4420 LET Q1=INT (Z1/8): LET S1=INT (Z1-8*Q1)
4430 LET Q2=INT (Z2/8): LET S2=INT (Z2-8*Q2)
4440 IF S1=0 THEN LET S1=8: LET Q1=Q1-1
4450 IF S2=0 THEN LET S2=8: LET Q2=Q2-1
4460 IF Q1<1 THEN LET Q1=1: GO SUB 4800
4470 IF Q2<1 THEN LET Q2=1: GO SUB 4800
4480 IF Q1>8 THEN LET Q1=8: GO SUB 4800
4490 IF Q2>8 THEN LET Q2=8: GO SUB 4800
4500 GO SUB 4900
4510 LET S1=INT (.5+S1): LET S2=INT (.5+S2): GO TO 2000
4800 GO SUB 9947: PRINT "DETENCION AUTOMATICA DE MOTORES";#0;AT 0,0;"PERMISO DEN
EGADO PARA ABANDONAR LA GALAXIA"
4830 RETURN
4900 LET E1=E1-N-10
4910 IF E1>=0 THEN GO TO 4960
4920 GO SUB 9947: PRINT "ENERGIA TRANSFERIDA DESDE EL ESCUDO PARA COMPLETAR MANI
OBRA"
4940 LET S0=S0+E1: IF S0<0 THEN LET S0=0
4950 LET E1=0
4960 LET TT1=.1*INT (10*W1): IF TT1>1 THEN LET TT1=1: LET T=T+TT1
4970 IF T>T0+T9 THEN GO TO 9994
4980 RETURN
5000 REM Disparo de canones laser
5010 IF K3>0 THEN GO TO 5040
5020 GO SUB 9947: PRINT : PRINT "NO HAY NAVES ENEMIGAS A LA VISTA": GO SUB 9950
5030 GO TO 3000
5040 IF F(4)>=0 THEN GO TO 5070
5050 GO SUB 9947: PRINT "EL CANON LASER NO FUNCIONA"
5060 GO TO 3000
5070 GO SUB 9947: PRINT "CANON LASER APUNTANDO OBJETIVO"
5090 PRINT "ENERGIA DISPONIBLE=";E1
5100 IF F(8)>=0 THEN GO TO 5120
5110 PRINT AT 16,44;"MAYOR PRECISION POR FALLO COMPUTADOR"
5120 INPUT "ENERGIA DISPARO:";C1: IF C1<0 OR C1>E1 THEN GO TO 5120
5140 LET E1=E1-C1
5150 GO SUB 9915
5160 IF F(7)>=0 THEN GO TO 5180
5170 LET C1=C1*RND
5180 LET C1=INT (C1/K3)
5190 FOR I=1 TO 3
5200 IF K(I,3)=0 THEN GO TO 5260
5210 LET H=INT ((2+RND)*C1/SQR (((K(I,1)-S1)*(K(I,1)-S1))+((K(I,2)-S2)*(K(I,2)-S
2))))
5220 IF H>=.15*K(I,3) THEN GO TO 5260
5230 LET K(I,3)=K(I,3)-H
5240 GO SUB 9947: PRINT H;" UNIDADES PARA ENEMIGO EN ";K(I,1);",";K(I,2)
5250 IF K(I,3)<0 THEN LET XA=K(I,1): LET XB=K(I,2): GO SUB 12200
5260 NEXT I: GO TO 2500
6000 REM Lanzamiento de un torpedo
6010 GO SUB 9947
6020 IF F(5)<0 THEN PRINT "LOS TUBOS DE TORPEDOS NO FUNCIONAN": GO TO 3000
6030 IF P1<=0 THEN PRINT "TORPEDOS AGOTADOS": GO TO 3000
6040 INPUT "CURSO TORPEDO (1-9) ";N1: IF N1<1 OR N1>9 THEN GO TO 6040
```

```
6060 IF N1=9 THEN LET N1=1
6070 LET X1=A(INT (N1))+B(INT (N1))*(N1-INT (N1))
6080 LET X2=C(INT (N1))+D(INT (N1))*(N1-INT (N1))
6090 LET E1=E1-2: LET P1=P1-1: LET XA=S1: LET XB=S2
6100 PRINT AT 21,0;"CURSO DEL TORPEDO: ";
6110 LET XA=XA+X1: LET XB=XB+X2
6120 IF (XA>=9) OR (XB>=9) OR (XA<1) OR (XB<1) THEN GO TO 6300
6130 PRINT AT 21,19;XA;",";XB
6140 LET I$=Q$(INT (XA+.5),2+4*INT (XB-.5))
6150 IF I$=" " THEN GO TO 6310
6160 IF I$="." THEN GO SUB 9890: GO TO 6250
6170 IF I$="*" THEN GO TO 6290
6180 GO SUB 9947: PRINT "BASE ESTELAR DESTRUIDA, ESTUPIDO"
6190 LET B3=B3-1: LET B9=B9-1: LET D0=0
6200 LET Q$(INT (XA),1+4*INT (XB-1))=" "
6210 LET Q$(INT (XA),2+4*INT (XB-1))=" "
6220 LET Q$(INT (XA),3+4*INT (XB-1))=" "
6230 LET G(Q1,Q2)=S3+10*(B3+10*K3)
6240 GO SUB 2020
6250 GO SUB 9915
6280 GO TO 2500
6290 GO SUB 9947: PRINT "UNA ESTRELLA ABSORBIO EL TORPEDO": GO TO 6250
6300 GO SUB 9947: PRINT "EL TORPEDO HA FALLADO EL OBJETIVO": GO TO 6250
6310 IF XH<>0 THEN PRINT AT 3+XH,XK: PRINT " "
6320 LET XH=INT (XA+.5): LET XK=2+4*INT (XB-.5)
6330 PRINT AT 3+XH,XK;"": BEEP 1,15: GO TO 6110
7000 REM Traspaso de energia al escudo protector
7010 GO SUB 9947
7020 IF F(7)>=0 THEN GO TO 7030
7025 GO SUB 9947: PRINT : PRINT "EL ESCUDO PROTECTOR NO FUNCIONA": GO TO 3000
7030 INPUT "ENERGIA AL ESCUDO ";N1: IF N1<0 OR N1>E1+S0 THEN GO TO 7030
7050 LET E1=E1+S0-N1
7060 LET S0=N1
7070 GO SUB 9947: PRINT : PRINT "ESCUDO A ";S0;" SEGUN SUS ORDENES"
7080 PRINT AT 19,26;"     ";AT 19,26;S0: GO SUB 9950: GO TO 3000
8000 REM Informe de danos
8005 FOR I=0 TO 10: PRINT AT I,0;"                                  ": NEXT I
8010 GO SUB 9947
8020 IF F(6)<0 THEN PRINT "EL CONTROL DE DANOS NO FUNCIONA": GO TO 8110
8030 PRINT AT 0,0;"CONTROL DE DANOS": FOR I=1 TO 8
8040 PRINT AT I,0;: GO SUB 8900: PRINT D$;F(I)
8050 NEXT I: GO SUB 9950
8110 IF D0=0 THEN GO TO 8300
8120 LET D3=0: FOR I=1 TO 8: IF F(I)<0 THEN LET D3=D3+.1
8130 NEXT I: IF D3>1 THEN LET D3=1
8140 IF D3=0 THEN GO TO 8300
8150 GO SUB 9947: PRINT "TECNICOS DISPUESTOS A REPARAR NAVE."
8160 PRINT #0;AT 0,0;"TIEMPO DE REPARACION ESTIMADO:";D3;"FECHAS ESTELARES": GO
SUB 9950
8180 GO SUB 9947: INPUT "AUTORIZA USTED LAS REPARACIONES S/N";I$: IF I$<>"SI" AN
D I$<>"NO" THEN GO TO 8180
8190 IF I$="NO" THEN GO TO 2500
8210 FOR I=1 TO 8: LET F(I)=0: NEXT I
8230 LET T=T+D3+.1: GO TO 8030
8320 FOR I=0 TO 10: PRINT AT I,0;"                                  ": NEXT I: GO
SUB 2020: GO TO 2500
8900 GO TO 8900+10*I
8910 LET D$="MOTORES ...........": RETURN
8920 LET D$="PANTALLA CORTO AL...": RETURN
8930 LET D$="PANTALLA LARGO AL...": RETURN
8940 LET D$="CANONES LASER.......": RETURN
8950 LET D$="TUBOS DE TORPEDOS...": RETURN
8960 LET D$="CONTROL DE DANOS....": RETURN
8970 LET D$="CONTROL DE ESCUDO...": RETURN
8980 LET D$="COMPUTADORA.........": RETURN
9000 REM Computadora de a bordo
9010 GO SUB 9947
9020 IF F(8)<0 THEN GO TO 9150
9030 PRINT "COMPUTADOR ACTIVO ESPERANDO ORDENES"
9050 PRINT #0;AT 0,0;"Q:Galaxia W:situacion E:torpedosR:base    T:dir-dist";
9060 GO SUB 9997: REM Lee una tecla
```

```
9070 IF I=CODE "Q" THEN GO TO 9200
9080 IF I=CODE "W" THEN GO TO 9400
9090 IF I=CODE "E" THEN GO TO 9500
9100 IF I=CODE "R" THEN GO TO 9700
9110 IF I=CODE "T" THEN GO TO 9800
9120 GO TO 9060
9150 PRINT "LA COMPUTADORA NO FUNCIONA": GO TO 3000
9200 REM Mapa de la galaxia
9205 FOR I=0 TO 9: PRINT AT I,0;"                                ": NEXT I
9210 PRINT AT 0,0;"MAPA ACUMULADO DE LA GALAXIA"
9220 PRINT AT 1,0;"   1  2  3  4  5  6  7  8"
9230 PRINT AT 2,0;"______________________________"
9240 FOR I=1 TO 8: PRINT AT 2+I,0;I;
9250 FOR J=1 TO 8: PRINT ("   "+STR$ (Z(I+1,J+1)))(2+(Z(I+1,J+1)>9)+(Z(I+1,J+1)>
99) TO 4+(Z(I+1,J+1)>9)+(Z(I+1,J+1)>99));: NEXT J: NEXT I
9260 PRINT AT 12,0;"PRESIONE UNA TECLA PARA SEGUIR"
9270 LET A$=INKEY$: IF A$="" THEN GO TO 9270
9330 CLS : GO SUB 2020: GO TO 2500
9400 REM Situacion
9410 PRINT AT 0,0;: FOR I=0 TO 10: PRINT "                                ";AT 0
,0;"INFORME DE SITUACION"
9420 PRINT : PRINT ;"Quedan ";K9;" naves enemigas"
9430 PRINT "Quedan ";T0+T9-T;" fechas estelares"
9440 PRINT "Hay ";B9;" bases estelares en la Galaxia"
9445 PRINT AT 10,0;"PULSE UNA TECLA PARA CONTINUAR": IF INKEY$="" THEN GO TO 944
5
9450 CLS : GO SUB 2020: GO SUB 2500
9500 REM Calculo de direcion para torpedos
9505 GO SUB 9947
9510 IF K3=0 THEN PRINT : PRINT "No hay naves enemigas aqui": GO SUB 9950: GO TO
 3000
9512 FOR I=1 TO 3
9515 IF K(I,3)=0 THEN GO TO 9525
9520 LET N1=K(I,1): LET N2=K(I,2): GO SUB 9530
9525 NEXT I: GO TO 3000
9528 REM Rutina de calculo de direccion/distancia
9530 PRINT "DIRECCION        DISTANCIA"
9540 LET X1=N1: LET X2=N2
9550 LET A=S1-X1: LET X=X2-S2
9560 IF X<0 THEN GO TO 9610
9562 IF A<0 THEN GO TO 9640
9570 IF X>0 THEN GO TO 9580
9572 IF A=0 THEN GO TO 9620
9580 LET C1=1
9590 IF ABS (A)>ABS (X) THEN LET C1=C1+2-ABS (X/A): GO TO 9600
9595 LET C1=C1+ABS (A/X)
9600 GO TO 9670
9610 IF A>0 THEN GO TO 9630
9615 IF X=0 THEN GO TO 9640
9620 LET C1=5: IF (A=0) AND (X=0) THEN GO TO 9670
9625 GO TO 9590
9630 LET C1=3: GO TO 9650
9640 LET C1=7
9650 IF ABS (A)<ABS (X) THEN LET C1=C1+2-ABS (A/X): GO TO 9670
9660 LET C1=C1+ABS (X/A)
9670 PRINT C1,SQR (X*X+A*A): GO SUB 9950: RETURN
9700 REM Calculo de direcion para bases
9710 GO SUB 9947
9720 IF B3=0 THEN GO TO 9790
9730 LET N1=I(1,1): LET N2=I(1,2): GO SUB 9530: GO TO 3000
9790 PRINT "No hay bases estelares aqui": GO SUB 9950: GO TO 3000
9800 REM Calculo de una direcion cualquiera
9810 INPUT "COORDENADA X ";X1: IF X1<1 OR X1>8 THEN GO TO 9810
9820 INPUT "COORDENADA Y ";X2: IF X2<1 OR X2>8 THEN GO TO 9820
9830 LET N1=X1: LET N2=X2: LET A=Q1-X1: LET X=X2-Q2
9870 GO SUB 9947: PRINT "DIRECCION        DISTANCIA"
9880 LET I=1: GO SUB 9560: GO SUB 9950: GO TO 3000
9890 REM Nave enemiga destruida
9892 FOR i=-10 TO 0: BEEP .1,i: NEXT i
9894 GO SUB 9947: PRINT : PRINT "NAVE ENEMIGA DESTRUIDA"
9896 LET K3=K3-1: LET K9=K9-1: IF K9=0 THEN GO TO 9910
```

```
9898 FOR L=1 TO 3: IF XA<>K(L,1) THEN GO TO 9900
9899 IF XB=K(L,2) THEN GO TO 9901
9900 NEXT L
9901 LET K(L,3)=0
9902 LET Q$(INT (XA+.5),1+4*INT (XB-.5))=" "
9903 LET Q$(INT (XA+.5),2+4*INT (XB-.5))=" "
9904 LET Q$(INT (XA+.5),3+4*INT (XB-.5))=" ": LET G(Q1,Q2)=S3+10*(B3+10*K3)
9905 GO SUB 2020: RETURN
9910 CLS : PRINT "Enhorabuena, capitan! La ultima nave enemiga"
9911 PRINT "ha sido destruida": PRINT
9912 PRINT "Su eficiencia es igual a";INT (1000*K7/(T-T0)): STOP
9915 REM Ataque enemigo
9916 IF K3=0 THEN RETURN
9917 IF D0=0 THEN GO TO 9919
9918 GO SUB 9947: PRINT "LA BASE PROTEGE LA NAVE AVENTURA": RETURN
9919 FOR L=1 TO 3: IF K(L,3)=0 THEN GO TO 9931
9920 LET H=SQR (((K(L,1)-S1)*(K(L,1)-S1))+((K(L,2)-S2)*(K(L,2)-S2)))
9921 LET H=INT ((2+RND)*K(L,3)/H)
9922 IF H=0 THEN GO TO 9931
9923 LET S0=S0-H: GO SUB 9947: PRINT H;" UNIDADES ALCANZAN AL AVENTU RA DESDE EL
 SECTOR ";K(L,1);",";K(L,2)
9925 IF S0<0 THEN CLS : PRINT "EL AVENTURA HA SIDO DESTRUIDO": GO TO 9994
9926 PRINT #0;AT 0,0;" ESCUDO DISMINUYE A ";S0
9927 IF (H<20) OR (.02>=H/S0) OR (RND>.5) THEN GO TO 9931
9928 LET I=INT (1+RND*8): LET F(I)=F(I)-(H/S0)-.5*RND
9929 BEEP .5,-10: BEEP .5,-15: GO SUB 9947: PRINT "CONTROL DE DANOS INFORMA:"
9930 GO SUB 8900: PRINT #0;AT 0,0;D$;"DANADO"
9931 GO SUB 9950: NEXT L: RETURN
9932 REM Reparaciones en marcha
9933 LET D1=14: LET D6=W1: IF D6>1 THEN LET D6=1
9934 FOR I=1 TO 8: IF F(I)=0 THEN GO TO 9939
9935 LET F(I)=F(I)+D6: IF F(I)<0 THEN GO TO 9939
9936 IF F(I)>0 THEN LET F(I)=0
9937 GO SUB 9947: PRINT #0;AT 0,0;: GO SUB 8900
9938 PRINT D$;" REPARADO ": GO SUB 9950
9939 NEXT I
9940 IF RND>.1 THEN RETURN
9941 LET I=1+INT (RND*8)
9942 LET F(I)=F(I)-.1-RND*5
9943 GO SUB 8900
9944 GO SUB 9947: PRINT "CONTROL DE DANOS INFORMA:"
9945 PRINT #0;AT 0,0;D$;" DANADO"
9946 RETURN
9947 REM Borra zona de informacion
9948 PRINT AT 20,0;"
   ";#0;AT 0,0;"
 ";
9949 PRINT AT 20,0;: RETURN
9950 REM Retardo
9951 FOR i=1 TO 40: LET j=2.2*i: NEXT i: RETURN
9994 PRINT "Es la fecha estelar ";t
9995 PRINT "Quedan ";K9;" naves enemigas en la Galaxia al final de su mision."
9996 PRINT "La federacion sera conquistada.": STOP
9997 LET A$=INKEY$: IF A$="" THEN GO TO 9997
9998 LET I=CODE (A$): RETURN
```

APENDICE 2





ALGUNOS JUEGOS DE ORDENADOR



Juegos inteligentes

1. Ajedrez. Existen muchos programas capaces de jugar aceptablemente al ajedrez. CHESS está escrito en BASIC, aunque su juego no es excesivamente bueno y a veces puede hacer jugadas sorprendentes, ajenas a las reglas normales del ajedrez. SARGON III tiene la particularidad de que sigue «pensando» sus jugadas mientras le toca el turno al jugador humano, lo que le permite alcanzar mayores profundidades de previsión, con lo que su juego es bastante potente. No se acerca, sin embargo, al nivel de un maestro del ajedrez.

2. Chaquete. Un programa (BKG 9.8), escrito para ordenador grande por Hans Berliner, logró derrotar en 1979 al campeón mundial de chaquete en un torneo a cinco partidas. Berliner es también autor de algunos de los mejores programas de ajedrez.

3. Nim. Este juego, originado en China hace muchos siglos, lo juegan dos oponentes. Se comienza con una serie de objetos dispuestos en filas, de la siguiente manera:

```
xxxxxxx
xxxxx
xxx
x
```

En cada turno, uno de los oponentes puede retirar los objetos que desee de una cualquiera de las filas. Tiene que retirar al menos un objeto. Gana el jugador que retire el último objeto. Existen muchas versiones para ordenador personal de este juego sencillo, que tiene una estrategia gana-

dora invencible para el jugador que comienza. (Una variante de NIM, jugado con cartas, aparece en la película «El año pasado en Marienbad».)

Un juego parecido al anterior es el de las 23 cerillas. Cada jugador, puede retirar, por turno, una, dos o tres cerillas del montón, que inicialmente tiene veintitrés. Pierde el que se queda con la última cerilla.

4. Awari. Se trata de un juego africano antiguo que se juega con siete palos y treinta y seis piedras dispuestas como sigue:

	000	000	000	000	000	000	
Casa							Casa del contrario
	000	000	000	000	000	000	

En cada turno, un jugador toma todas las piedras de una casilla en su propio lado y las va distribuyendo por las casillas contiguas en el sentido contrario a las agujas del reloj. Si la última piedra queda en su casa, el jugador puede repetir turno. Si la última piedra cae en una casilla vacía y la casilla opuesta no está vacía, las piedras en la casilla opuesta y la última que se colocó pasan a la casa del jugador. Cuando uno de los dos lados se queda vacío, gana el jugador que tiene más piedras en su casa.

Existen también muchas versiones de este juego para ordenadores personales.

5. Hexapawn. Este juego, inventado por Martin Gardner, se juega con peones de ajedrez en un tablero de 3 por 3 casillas. Los peones se mueven como en ajedrez: un espacio hacia adelante para avanzar, uno en diagonal para capturar una pieza contraria. En la posición inicial, los tres peones de cada jugador están en la primera fila del tablero, la más próxima a él. El objetivo del juego es llevar un peón hasta la fila opuesta (la fila del contrario) o conseguir que éste no pueda mover ninguna pieza cuando le corresponda jugar. Existen programas sencillos que aprenden por «experiencia» cuáles son las mejores jugadas, y que llegan a ser invencibles después de cierto número de partidas.

6. Mastermind. Este conocido juego tiene por objeto adivinar una combinación de colores elegida por el contrario. El jugador propone una combinación determinada, y el contrario le dice cuántos colores están en su sitio y cuántos están presentes, pero en un lugar diferente. Por ejemplo: si la combinación oculta es

Rojo Verde Amarillo Rojo

y el jugador adivinó:

Rojo Marrón Azul Amarillo

la respuesta es: un color está en su sitio, y otro fuera de él.

Es fácil construir programas que preparen una combinación de colores y permitan que el jugador humano la adivine en cierto número de in-

tentos. Algo más complicado (pero también existen varias versiones) es conseguir que el programa adivine nuestra combinación.



Juegos de azar

1. El juego CASINO permite elegir entre probar suerte con una máquina tragaperras, o jugar al póquer o al Blackjack (versión inglesa de las siete y media, donde cada carta tiene el valor correspondiente a su número, excepto las figuras, que valen diez puntos, y el as, para el que se puede elegir el valor de uno u once puntos, a voluntad. Gana quien obtenga 21 puntos, o se acerque a dicho valor más que el contrario. Pierde inmediatamente quien se pasa de 21 puntos.

2. Pin-ball. Se trata de un programa que imita hasta extremos increíbles (con ayuda de gráficos) el funcionamiento de una de las máquinas «pin-ball» o del «millón». Los rebotes de las bolas en los obstáculos están muy conseguidos. Para controlar los elementos móviles se utilizan teclas.



Juegos de habilidad

1. Decatlón. Se trata de un juego complejo, con gráficos bastante espectaculares. Los jugadores se turnan en cada una de las pruebas, en las que deben utilizar los dedos para simular los ejercicios atléticos correspondientes. Por ejemplo, en las pruebas de carrera se hace uso de dos dedos que tocan alternativamente dos teclas determinadas (si la carrera es en línea recta) o cuatro (si es preciso girar en las curvas del circuito). En las pruebas de salto se usan diferentes teclas para correr, tomar impulso y saltar. Hay también pruebas de lanzamiento de jabalina, donde cierta tecla sirve para doblar el brazo, otra para girarlo y otra para dar impulso y soltar el proyectil. En cada una de las pruebas se le proporciona a los jugadores el número de puntos que han conseguido, de acuerdo con las reglas del decatlón.

2. Baloncesto. En las versiones más sencillas, se da a elegir al jugador entre varias jugadas posibles, prefabricadas. El programa calcula, dependiendo de su posición y de la jugada elegida, si el jugador ha encestado. A continuación, el programa elige también su jugada y proporciona el resultado. Como era de esperar, gana el que consiga más encestes.

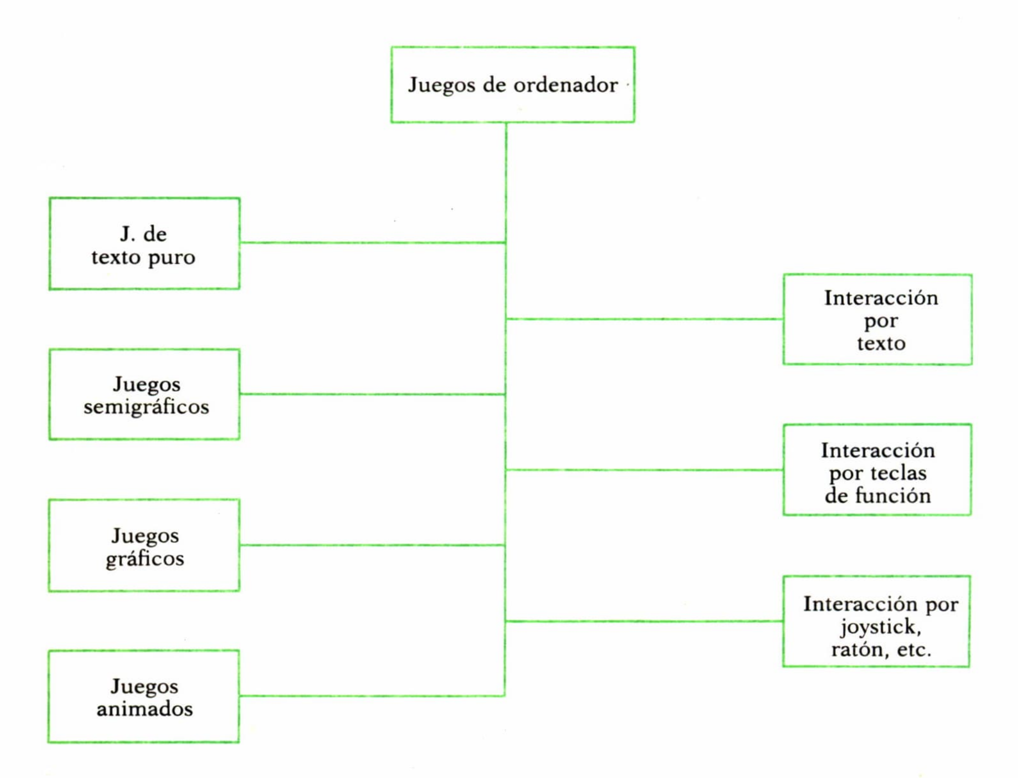
3. Bolos. El programa presenta la disposición de los bolos y el jugador debe elegir el modo de tiro de la pelota. Después de cada jugada, el programa presenta la disposición en que han quedado los bolos.

4. Tenis de mesa. Hay versiones para uno o dos jugadores. El jugador dispone de una raqueta (que aparece dibujada sobre la pantalla) que

puede mover mediante teclas. Es posible controlar la velocidad de la pelota, para hacer el juego más o menos difícil.

5. Frogger. Se trata de un juego gráfico en el que el jugador lleva el control de los movimientos de una rana que desea cruzar la calzada de una calle y un canal de agua para alcanzar una mosca que se encuentra al otro lado. En la calzada, la rana debe evitar que le alcancen los coches (que vienen a diferentes velocidades) y en el canal debe evitar caerse al agua (donde se ahogaría), saltando de tronco en tronco o aprovechando el paso de una tortuga. Existen también enemigos que pueden comerse a la rana, si la alcanzan (un cocodrilo, un tiburón, una serpiente). El juego tiene varios niveles de dificultad, y al terminar uno correctamente se para automáticamente al siguiente.

6. J-Bird. El jugador controla los movimientos de un pajarillo que debe saltar de casilla en casilla hasta conseguir pasar por todas antes de que le capture uno de sus enemigos: gatos, serpientes, bolas más o menos grandes que caen aleatoriamente y que pueden aplastarle. Existen varios niveles de dificultad.



Clasificación de los juegos según la forma de interaccionar con el jugador.

7. STYT. En este juego, el jugador controla mediante las teclas un punto que se mueve en el perímetro de un rectángulo. En el interior de éste hay un flagelo móvil muy rápido, que barre continuamente su superficie con numerosos y bruscos cambios de dirección. Otro enemigo es un gusano que se mueve también por el perímetro y que intenta alcanzar y destruir el punto controlado por el jugador. Este puede abandonar el perímetro e introducirse en el rectángulo, encerrar una zona de éste a lo largo de su trayectoria y volver de nuevo al perímetro antes de que le alcance el flagelo. La zona abarcada por la trayectoria seguida por el jugador queda coloreada, y a partir de ese momento el flagelo no puede penetrar en ella. Se trata de irle robando terreno poco a poco, hasta dejarle menos de un 20 por ciento del rectángulo original. Cuando se llega a esta meta, se pasa al nivel siguiente, donde el jugador deberá enfrentarse con dos gusanos que se mueven en direcciones opuestas, y así sucesivamente.

8. Bigtop. Se trata de un juego gráfico animado, versión gráfica del comecocos, en el que el jugador controla un muñeco que puede saltar, deslizarse por una barra, etc. A lo largo de sus movimientos debe adquirir cierto número de objetos (cada uno de los cuales vale puntos). Cuando los ha obtenido todos, aparece una escalera que le permite ascender al nivel siguiente. A lo largo de sus andanzas, el muñeco se encuentra con enemigos a los que debe evitar, pelotas gigantes que pueden aplastarlo, individuos que le arrojan objetos que, si le dan, le descalabran, etc. También debe evitar caerse desde las alturas. En cada partida, el jugador dispone de cierto número de vidas, agotadas las cuales el juego termina.

Juegos educativos

1. Hammurabi. De autor desconocido, este fue uno de los primeros juegos para ordenador personal que alcanzó gran popularidad. El jugador debe gobernar una ciudad, o un país. El programa le ofrece los datos sobre los recursos de que dispone (territorio arable, semillas disponibles, población) y le pide que distribuya la semilla (para alimento, para sembrar, para guardar, para comprar más tierra). La población crece de forma natural (aunque también pueden producirse epidemias que la diezmen). Las cosechas son buenas o malas, de forma aleatoria. Las ratas se comen parte de las semillas almacenadas. Como consecuencia de todos estos factores, el jugador descubre que no es nada fácil controlar el país durante los diez años que dura su mandato. Al final, el programa le proporciona un juicio sobre su eficacia administrativa.

2. Adivinar palabras o números. Existen numerosas versiones de estos juegos sencillos. Uno de los más conocidos es el «ahorcado», con el que también se puede jugar con papel y lápiz.

ENCICLOPEDIA PRACTICA DE LA

INFORMATICA APLICADA

INDICE GENERAL

5 PRACTIQUE MATEMATICAS Y ESTADISTICA CON EL ORDENADOR

En este libro se repasan los principales conceptos de las Matemáticas y la Estadística, desde un punto de vista eminentemente práctico y para su aplicación al ordenador personal. Se basan los diferentes textos en la presentación de pequeños programas (que usted podrá introducir en su ordenador personal).

6 AÑADA PERIFERICOS A SU ORDENADOR

Breve descripción de varios periféricos que facilitan la comunicación con el ordenador personal, con algunos ejemplos de fácil construcción: ratón, lápiz óptico, marco para pantalla táctil...

7 APL: LENGUAJE PARA PROGRAMADORES DIFERENTES

APL es un lenguaje muy potente que proporciona gran simplicidad en el desarrollo de programas y al mismo tiempo permite programar sin necesidad de conocer todos los elementos del lenguaje. Por ello es ideal para quienes reúnan imaginación y escasa formación en Informática.

8 DISPOSITIVOS INTERACTIVOS PARA SU ORDENADOR

Descripción detallada de la forma de construir, paso a paso y en su propia casa, dispositivos electrónicos que aumentarán la potencia y facilidad de uso de su ordenador: tableta digitalizadora, convertidores de señales analógicas, comunicaciones entre ordenadores.

9 CRIPTOGRAFIA: LA OCULTACION DE MENSAJES Y EL ORDENADOR

En este libro se presentan las técnicas de ocultacion de mensajes a través de la criptografía desde los primeros tiempos hasta la actualidad, en que el uso de los computadores ha proporcionado la herramienta necesaria para llegar al desarrollo de esta ciencia.

10 PRACTIQUE CIENCIAS NATURALES CON EL ORDENADOR

Ejemplos sencillos para practicar con el ordenador. Casos curiosos de la Naturaleza en forma de programas para su ordenador personal.

11 GRAFICOS ANIMADOS CON EL ORDENADOR

En este libro las técnicas utilizadas para la animación son el resultado de unas pocas ideas básicas muy sencillas de comprender. Descubrirá los trucos y secretos de movimientos, choques, rebotes, explosiones, disparos, saltos, etc.

12 JUEGOS INTELIGENTES EN MICROORDENADORES

Los ordenadores pueden enfrentarse de forma «inteligente» ante puzzles y otros tipos de juegos. Esto es posible gracias al nuevo enfoque que ha dado la IA a la tradicional teoría de juegos.

13 ECONOMIA DOMESTICA CON EL ORDENADOR PERSONAL

Breve introducción a la contabilidad de doble partida y su aplicación al hogar, con explicaciones de cómo utilizar el ordenador personal para facilitar los cálculos, mediante un programa especialmente diseñado para ello.

14 COMO SIMULAR CIRCUITOS ELECTRONICOS EN EL ORDENADOR

Introducción a los diferentes métodos que se pueden emplear para simular y analizar circuitos electrónicos, mediante la utilización de diferentes lenguajes.

15 LOS LENGUAJES DE LA INTELIGENCIA ARTIFICIAL

Libro en que se describen los lenguajes específicos para la «elaboración del saber» y los entornos de programación correspondientes. El conocimiento de estos lenguajes, además de interesante en sí mismo, es sumamente útil para entender todo lo que la Informática Artificial supondrá para el futuro de la Informática.

16 PRACTIQUE FISICA Y QUIMICA CON SU ORDENADOR

Libro eminentemente práctico para realizar pequeños «experimentos» con su ordenador y distraerse de un modo útil.

17 EL ORDENADOR Y LA LITERATURA

En este libro se examinan procesadores de textos, programas de análisis literario y una curiosa aplicación desarrollada por el autor: APOLO, un programa que compone estructuras poéticas.

18 COMO ELEGIR UNA HOJA ELECTRONICA DE CALCULO

En este título se estudian las diferentes versiones existentes de esta aplicación típica, desde el punto de vista de su utilidad para, en función de las necesidades de cada usuario y del ordenador de que dispone, poder elegir aquella que más se adecúe a cada caso.

19 DIBUJOS TRIDIMENSIONALES EN EL ORDENADOR PERSONAL

Compruebe que también con su ordenador personal puede llegar a diseñar y calcular imágenes en tres dimensiones con técnicas semejantes a las utilizadas por los profesionales del dibujo con equipos mucho más sofisticados.

20 ¿MAQUINAS MAS EXPERTAS QUE LOS HOMBRES?

Después de situar los «sistemas expertos» en el contexto de la inteligencia artificial y describir su construcción, su funcionamiento, su utilidad, etc., se analiza el papel que pueden tener en el futuro (y presente, ya) de la Informática.

21 PRACTIQUE HISTORIA Y GEOGRAFIA CON SU ORDENADOR
Libro interesante para los aficionados a estas ciencias, a quienes presenta una nueva visión de cómo utilizar el microordenador en su estudio.

22 ERGONOMIA: COMUNICACION EFICIENTE HOMBRE-MAQUINA
Análisis de la comunicación entre el hombre y la máquina, y estudio de diferentes soluciones que tienden a facilitarla lo más posible.

23 EL ORDENADOR Y LA ASTRONOMIA
Los cálculos astronómicos y el conocimiento del firmamento en un libro apasionante y curioso.

24 VISION ARTIFICIAL. TRATAMIENTO DE IMAGENES POR ORDENADOR
El procesado de imágenes es un campo de reciente y rápido desarrollo con importantes aplicaciones en áreas tan diversas como la mejora de imágenes biomédicas, robóticas, teledetección y otras aplicaciones industriales y militares. Se presentan los principios básicos, los sistemas y las técnicas de procesado más usuales.

25 LA ESTACION TERMINAL PERSONAL
Las modernas técnicas de comunicación van permitiendo que las grandes capacidades de proceso y el acceso a bases de datos de gran tamaño estén cada día más al alcance de cada usuario (fuera ya de los Centros de Proceso de Datos).

26 EL ORDENADOR COMO MAQUINA DE ESCRIBIR INTELIGENTE
Descripción de los sistemas de tratamiento de textos existentes, análisis comparativos y estudio de posibilidades de cada uno de ellos. Guía práctica para la elección del presente paquete que más se adecúe a nuestras necesidades y al ordenador personal de que dispongamos.

27 EL LENGUAJE C, PROXIMO A LA MAQUINA
Lenguaje de programación que se está imponiendo en los microordenadores más grandes, tanto por su facilidad de aprendizaje y uso, como por su enorme potencia y su adecuación a la programación estructurada. Vinculado íntimamente al sistema operativo UNIX es uno de los lenguajes de más futuro entre los que utilizan los micros personales.

28 EL ORDENADOR COMO INSTRUMENTO MUSICAL Y DE COMPOSICION

Análisis de cómo se puede utilizar el ordenador para la composición o interpretación de música. Libro eminentemente práctico, con numerosos ejemplos (que usted podrá practicar en su ordenador casero) y lleno de sugerencias para disfrutar haciendo de su ordenador un verdadero instrumento musical.

29 LA CREATIVIDAD EN EL ORDENADOR. EXPERIENCIAS EN LOGO

El LOGO es un lenguaje enormemente capacitado para la creación principalmente gráfica y en especial para los niños. En este sentido se han desarrollado numerosas experiencias. En el libro se analizan estas experiencias y las posibilidades del LOGO en este sentido, así como su aplicación a su ordenador casero para que usted mismo (o con sus hijos) pueda repetirlas.

30 SISTEMAS OPERATIVOS: EL SISTEMA NERVIOSO DEL ORDENADOR

Características de diversos sistemas operativos utilizados en los ordenadores personales y caseros. Se trata de llegar al conocimiento, ameno, aunque riguroso, de la misión del sistema operativo de su ordenador, para que usted consiga sacar mayor rendimiento a su equipo.

NOTA: Ediciones Siglo Cultural, S. A., se reserva el derecho de modificar, sin previo aviso, el orden, título o contenido de cualquier volumen de la colección.

La proliferación de los juegos para computadora ha sido una de las causas principales del gran auge de los ordenadores caseros en nuestra década. Entre las muchas clases de juegos que existen, este libro se dedica especialmente a los de aventura, que permiten al jugador identificarse (sin correr ningún riesgo personal) con un héroe que busca un tesoro, con el capitán de una nave espacial en una escena de guerra estelar, o con uno de los grandes guerreros de la historia. De todos estos tipos de juego hay ejemplos en el libro, que culmina con una aplicación que permite al lector escribir sus propias escenas de multiaventura.